UNIVERSIDAD CARLOS III DE MADRID

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

TRABAJO FIN DE GRADO

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DISEÑO DE ALGORITOMO DE DETECCIÓN DE OBSTÁCULOS EN ENTORNO VIARIO BASADO EN VISIÓN POR COMPUTADOR

Autor: Roberto Monte Morgado

Director: Aurelio Ponz Vila

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En la actualidad, millones de personas usan el automóvil como medio de transporte para sus desplazamientos diarios, y por desgracia como todos sabemos, también es la causa de miles de accidentes con víctimas. A lo largo de la existencia del automóvil siempre ha sido una prioridad el intentar reducir las víctimas, y según ha ido pasando el tiempo, y con ello el mayor número de usuarios de automóviles, estas medidas de seguridad se han visto incrementadas.

España presenta una tasa de 36 muertos por millón de habitante con la cual se supera con antelación la marcada en el Plan Estratégico para 2020.

Según fuentes de la DGT, en la siniestralidad de 2014 destacan las siguientes circunstancias:

- Se ha constatado un aumento de 5,5 millones de viajes de largo recorrido por carretera. Ese aumento se eleva hasta el 2,5% si nos referimos a viajes en fines de semana.
- Mayor envejecimiento del parque. Los vehículos en que hubo fallecidos tenían una edad media de 12,3 años los turismos y 11,8 las furgonetas.
- Se han detectado más infracciones entre conductores de furgonetas y en camiones de transporte sobre todo en este último tipo de vehículos por tiempos de conducción y descanso. El número de víctimas mortales en furgonetas ha pasado de 47 en 2013 a 92 en 2014.
- También más infracciones por consumo problemático de drogas ilegales. Este año la DGT ha realizado más de 24.000 pruebas, con un resultado de 25% positivas. Para el 2015 la DGT pretende llegar a realizar la prueba a los aproximadamente 90.000 conductores implicados en accidentes.
- Se mantiene la continuada reducción en vías interurbanas de alta capacidad, con 35 muertos menos que en 2013. Al contrario, aumentan las víctimas mortales en vías convencionales, con 32 muertos más.
- Persiste un núcleo de usuarios que continua sin utilizar los elementos de seguridad. En 2014 murieron 131 personas en turismos y 23 en furgonetas que no usaban cinturón de seguridad. Y otros 14 sin casco en bicicletas y 2 en motocicletas.

1.1. Evolución histórica.

La evolución de las cifras de fallecidos por accidente de tráfico con víctimas, desde que se mantienen estadísticas, muestra a partir del año 1960 una tendencia general ascendente hasta alcanzar un máximo en el año 1989, en el que se notificaron 9.344

fallecidos. Desde entonces, el número de fallecidos ha ido disminuyendo de manera más o menos acusada hasta alcanzar el mínimo documentado en 2013, con 1.680 fallecidos.

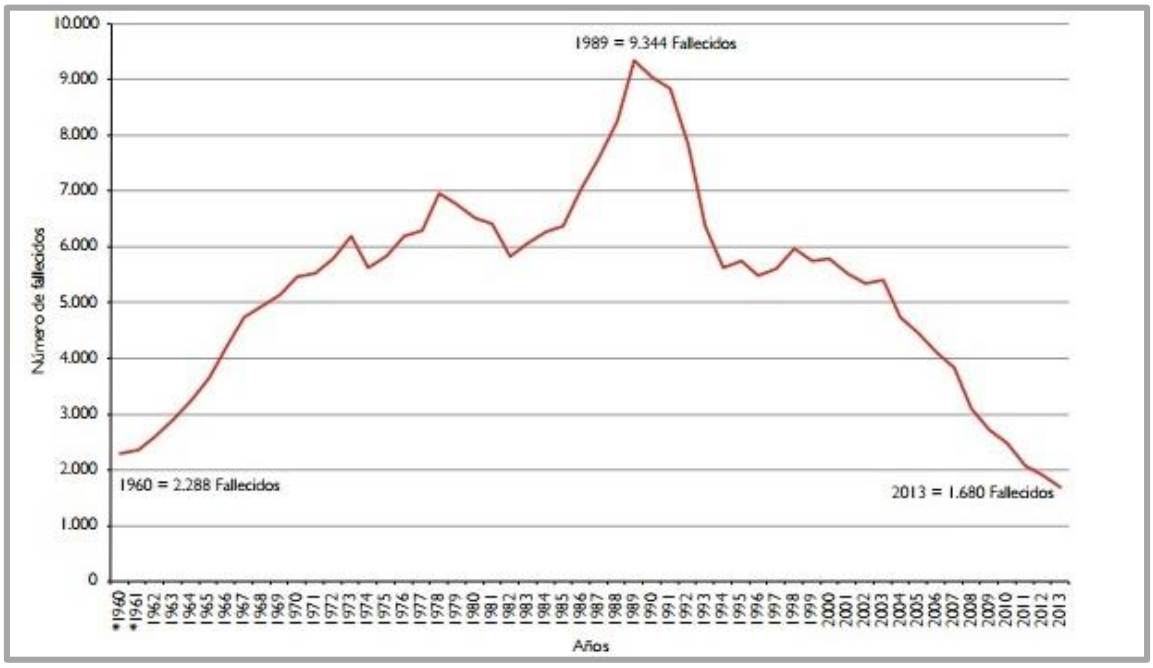


Figura 1: Evolución de los fallecidos en accidente de tráfico con víctimas. Serie 1960-2013.

La letalidad, definida como la razón entre el número de fallecidos y el número de víctimas, ha disminuido desde 1993, no sólo debido a la disminución de los fallecidos, sino también al aumento de los registros de heridos leves. En 2013 los heridos leves suponen el 91% de las víctimas registradas, mientras que en 1993 eran el 65%. En esta figura se observa el descenso sostenido del índice de letalidad, que fue de 1,3 en este último año, inferior al detectado el año anterior.

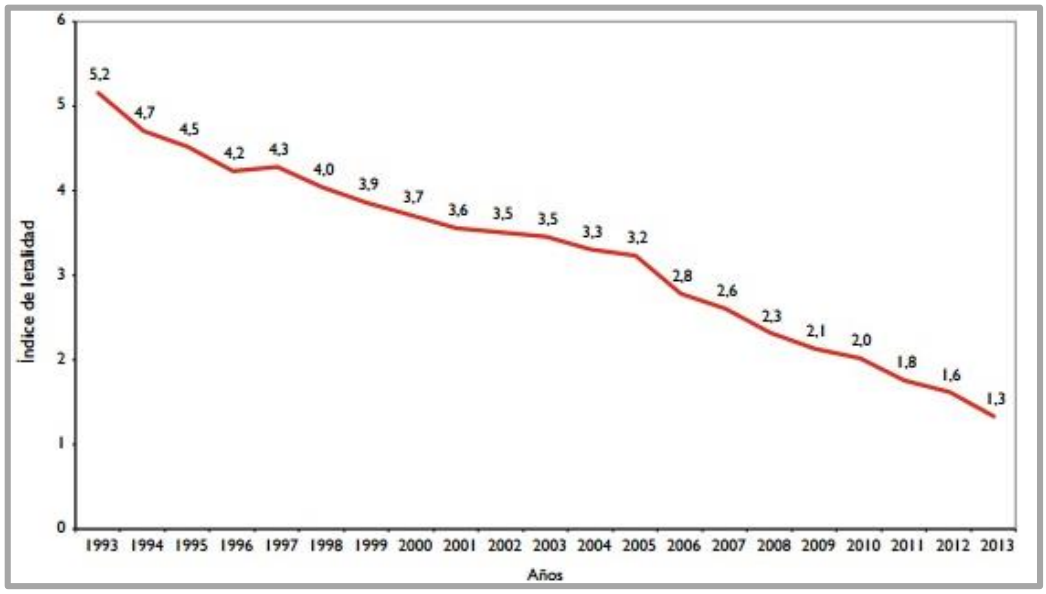


Figura 2: Evolución de la letalidad (n.º de fallecidos/ n.º de víctimas x 100) en los accidentes de tráfico con víctimas. Serie 1993–2013.

1.2. El lugar del accidente.

Durante 2013, la mayoría de los accidentes con víctimas tuvo lugar en vías urbanas, localizándose seis de cada diez accidentes en este tipo de vías. No obstante, las lesiones mortales se concentran en vías interurbanas, donde el número de fallecidos es casi tres veces superior al de vías urbanas. Por lo que respecta al número de heridos graves, en 2013 un 51% de los casos se produjeron en vías interurbanas, y un 49% en vías urbanas.

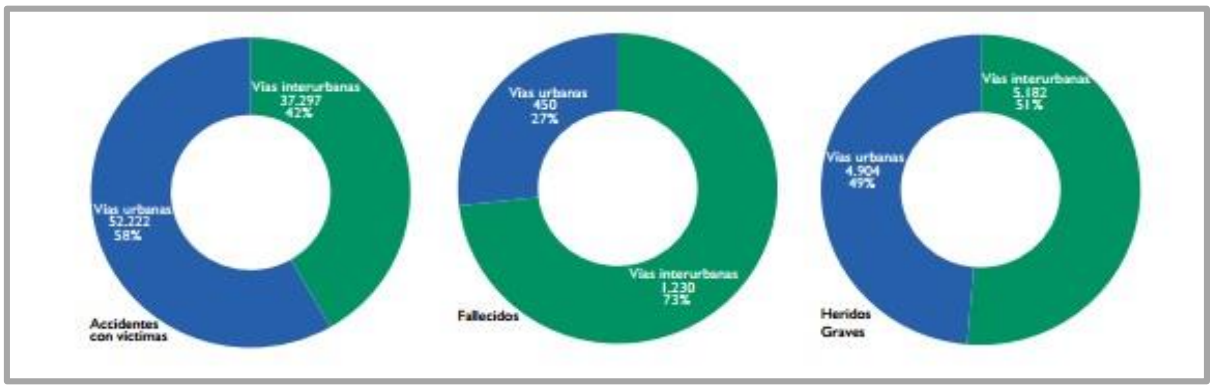


Figura 3: Distribución del número de accidentes con víctimas, fallecidos y heridos graves según zona. Año 2013.

Durante los últimos diez años se han observado reducciones del número de fallecidos y heridos graves tanto en vías interurbanas como en vías urbanas, si bien la magnitud de la reducción ha sido menor en estas últimas.

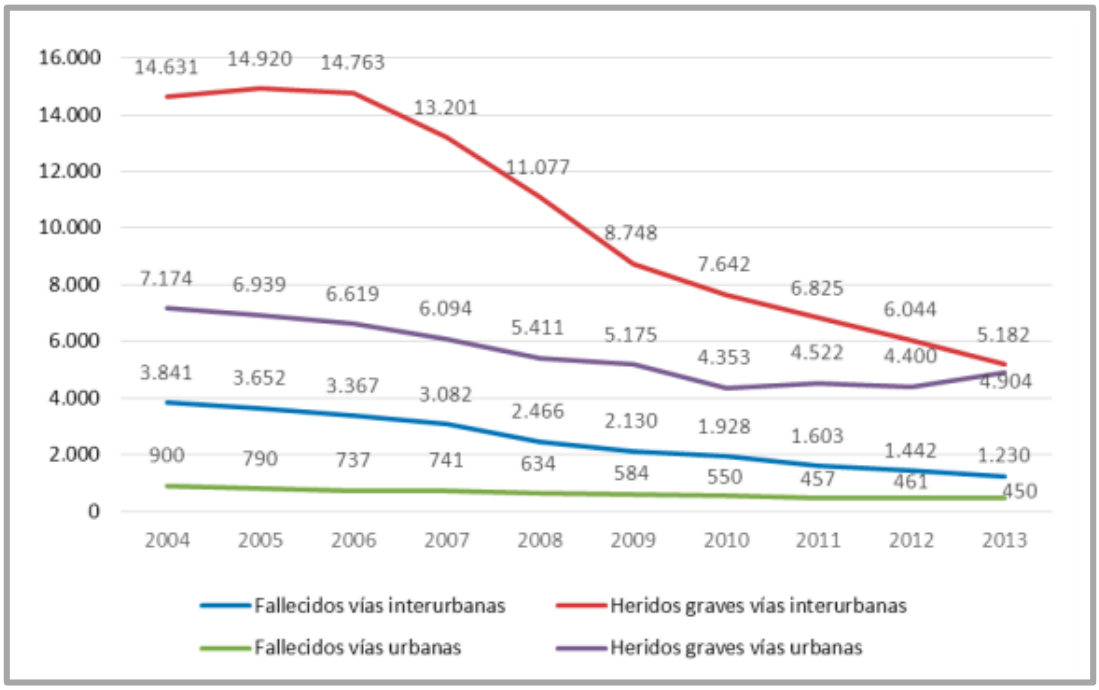


Figura 4: Evolución de los fallecidos y heridos graves en vías interurbanas y urbanas, 2004-2013.

1.2.1. Las vías interurbanas.

En el año 2013, el 42% de los accidentes de tráfico con víctimas se registraron en las vías interurbanas, alcanzando la cifra de 37.297 accidentes, y en ellos se produjeron el 73% de las víctimas mortales, 1.230 fallecidos. Siguiendo la evolución decreciente que se viene observando a lo largo de estos años, los fallecidos por accidente de tráfico en las vías interurbanas descendieron un 15% en 2013 respecto del año anterior y los heridos graves lo hicieron en un 14%. En cuanto a los heridos leves, se ha producido un aumento del 7%.

El índice de letalidad en 2013 en las vías interurbanas fue de 2,1, un valor superior a la letalidad para el conjunto de las vías que fue de 1,3.

Vías interurbanas	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	Variación 2013/2012	Variación Interanual 2004-2013
Accidentes con víctimas	43.787	42.624	49.221	49.820	43.831	40.789	39.174	35.878	35.425	37.297	5%	-2%
Fallecidos	3.841	3.652	3.367	3.082	2.466	2.130	1.928	1.603	1.442	1.230	-15%	-12%
Heridos graves	14.631	14.920	14.763	13.201	11.077	8.748	7.642	6.825	6.044	5.182	-14%	-11%
Heridos leves	56.459	53.869	62.306	63.587	56.222	54.180	52.247	47.692	47.936	51.320	7%	-1%
Índice de letalidad	5,1	5,0	4,2	3,9	3,5	3,3	3,1	2,9	2,6	2,1		
Tráfico vehículo-km 10^{6}	241.715	245.073	247.877	256.660	251.749	249.371	241.131	236.065	224.285	221.610	-1%	-1%
Fallecidos por cien millones vehículo km	1,6	1,5	1,4	1,2	1,0	0,9	0,8	0,7	0,6	0,6		

Figura 5: Evolución de los accidentes con víctimas en vías interurbanas, 2004-2013.

1.2.2. Las vías urbanas.

En el año 2013, las vías urbanas registraron un total de 52.222 accidentes con víctimas, en los cuales fallecieron 450 personas (el 27% del total), 4.904 resultaron heridas graves y 63.314 heridas leves. Respecto del año anterior, el número de accidentes con víctimas ha crecido un 10%, el de heridos graves un 11% y el de heridos leves un 10%. El número de fallecidos ha disminuido un 2%.

El índice de letalidad para las vías urbanas en 2013 fue 0,7, igual al del año anterior.

Vías urbanas	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	Variación 2013/2012	Variación Interanual 2004-2013
Accidentes con víctimas	50.222	48.563	50.576	50.688	49.330	47.462	46.329	47.149	47.690	52.222	10%	0%
Fallecidos	900	790	737	741	634	584	550	457	461	450	-2%	-7%
Heridos graves	7.174	6.939	6.619	6.094	5.411	5.175	4.353	4.522	4.400	4.904	11%	-4%
Heridos leves	60.119	57.081	59.762	59.639	58.237	56.863	56.103	56.588	57.510	63.314	10%	1%
Índice de letalidad	1,3	1,2	1,1	1,1	1,0	0,9	0,9	0,7	0,7	0,7		

Figura 6: Evolución de los accidentes con víctimas en vías urbanas, 2004-2013.

1.3. Objetivo.

El objetivo del presente proyecto es el desarrollo de un algoritmo que permita detectar obstáculos, en este caso coches, en el entorno de un vehículo que circula por una zona urbana compleja mediante un sistema de visión instalado en el interior del vehículo. Se busca que el tiempo de computación sea el menor posible, ya que se trabaja con requisitos de tiempo real y cuanto antes se obtengan los resultados, antes podrá actuar el conductor ante una posible colisión.

El algoritmo ha sido desarrollado en los lenguajes de programación C/C++, basándose en las librerías OpenCV en el entorno de desarrollo integrado QtCreator, en el sistema operativo Ubuntu Linux.

2. ESTADO DEL ARTE

El Estado del Arte del presente proyecto se centra en los sistemas de seguridad que presentan los vehículos que se encuentran en el mercado y aquellos que se están desarrollando actualmente para reducir el índice de mortalidad existente en las carreteras.

2.1. Seguridad en el automóvil.

Los fabricantes de automóviles han trabajado durante años para conseguir mejorar sus vehículos en materia de seguridad vial. Actualmente, la seguridad activa y la seguridad pasiva funcionan en los vehículos con el fin de proteger la vida del conductor.

Los fabricantes adaptan las nuevas tecnologías en función de las normas dictadas por organismos internacionales que realizan investigaciones sobre las causas de los accidentes de tráfico. La finalidad última es mejorar la seguridad vial, protegiendo la vida del conductor y los acompañantes.

2.1.1. Seguridad pasiva.

Son los elementos que tratan de minimizar los posibles daños de los integrantes del vehículo en el caso de que el accidente sea inevitable, es decir, no evitan el accidente pero sí reducen sus consecuencias. Dentro de este grupo se engloban los cinturones de seguridad, los airbags, los reposacabezas, los cristales así como el chasis y la carrocería capaces de absorber la mayor energía posible en el impacto.

2.1.2. Seguridad activa.

Con el progreso de la tecnología de estos últimos años es posible encontrar una solución a los factores humanos, que son los principales causantes de accidentes hoy en día, gracias al uso de elementos electrónicos y programas informáticos. La seguridad activa integra todos aquellos elementos que incorporan los vehículos destinados a disminuir el riesgo de que se produzca un accidente. Entre ellos se pueden distinguir los sistemas de frenado, dirección y suspensión con control electrónico, la iluminación, los amortiguadores y los neumáticos. Los sistemas que los fabricantes han desarrollado en las últimas décadas para mejorar la seguridad activa son el antibloqueo de frenos y ruedas (ABS), la tracción total o los controles de estabilidad (ESP) y tracción. Con la

correcta utilización y conservación de estos elementos pueden reducirse en gran medida los accidentes y sus posibles consecuencias.

2.2. Sistemas avanzados de asistencia al conductor (ADAS).

Los sistemas ADAS (*Advanced Driver Assistance Systems*) tratan de desarrollar tecnologías inteligentes aplicadas al automóvil con el propósito tanto de aumentar la seguridad en las carreteras consiguiendo reducir el número de accidentes, como de facilitar la conducción para que ésta sea más confortable.

La implementación de estos sistemas en los vehículos es reciente, aunque existe un incremento de manera muy importante en la actualidad. Se podría considerar que el primer sistema ADAS integrado en un vehículo de serie fue el "Adaptative Cruise Control" en 1995, y que fue incorporado en un vehículo para el mercado japonés. Este dispositivo consiste básicamente en controlar la velocidad del vehículo según la distancia de seguridad con el vehículo precedente. En Europa se podía ver por primera vez en 1999 en la Clase S de Mercedes-Benz. Estos primeros dispositivos se empezaron a instalar solamente en coches de gama alta y a precios elevados, pero en la actualidad cada vez es más común que se integren en vehículos de clase media, y se espera que en los próximos años la instalación tenga una crecida casi exponencial, ayudando a reducir los accidentes de tráfico y las consecuencias que estos provocan sobre las personas.

A continuación se explicaran algunos de los sistemas ADAS que podemos encontrar a día de hoy en los vehículos, o que se estén desarrollando actualmente.

2.2.1. Control de crucero adaptativo (ACC).

Durante los viajes largos, especialmente en autopistas y autovías, la conducción puede hacerse especialmente rutinaria y pesada puesto que el conductor para adaptarse constantemente a las situaciones, prácticamente sólo debe acelerar o frenar su vehículo.

Por ello existen los sistemas de control de crucero que nos permiten mantener una velocidad constante previamente fijada por el conductor y que se desconectará automáticamente si pisamos el freno.

El control de crucero adaptativo (ACC, *Adaptative Cruise Control*) es una evolución del anterior y ahora se muestra más inteligente y tiene en cuenta el tráfico a la hora de mantener la velocidad, con lo que el conductor no tendrá que estar acoplándose continuamente a las condiciones de la carretera.

El control de crucero adaptativo cuenta con una serie de sensores que se encargan de detectar el tráfico en la vía, de tal manera que si nos encontramos con un coche por

delante a una velocidad inferior, automáticamente el sistema alerta al conductor del peligro y reduce la velocidad de nuestro vehículo actuando sobre el sistema de frenos, de forma que se mantiene la distancia de seguridad que haya sido predeterminada. Una vez que el carril por el que circulamos queda libre, el sistema acelera el vehículo hasta la velocidad que hayamos programado.

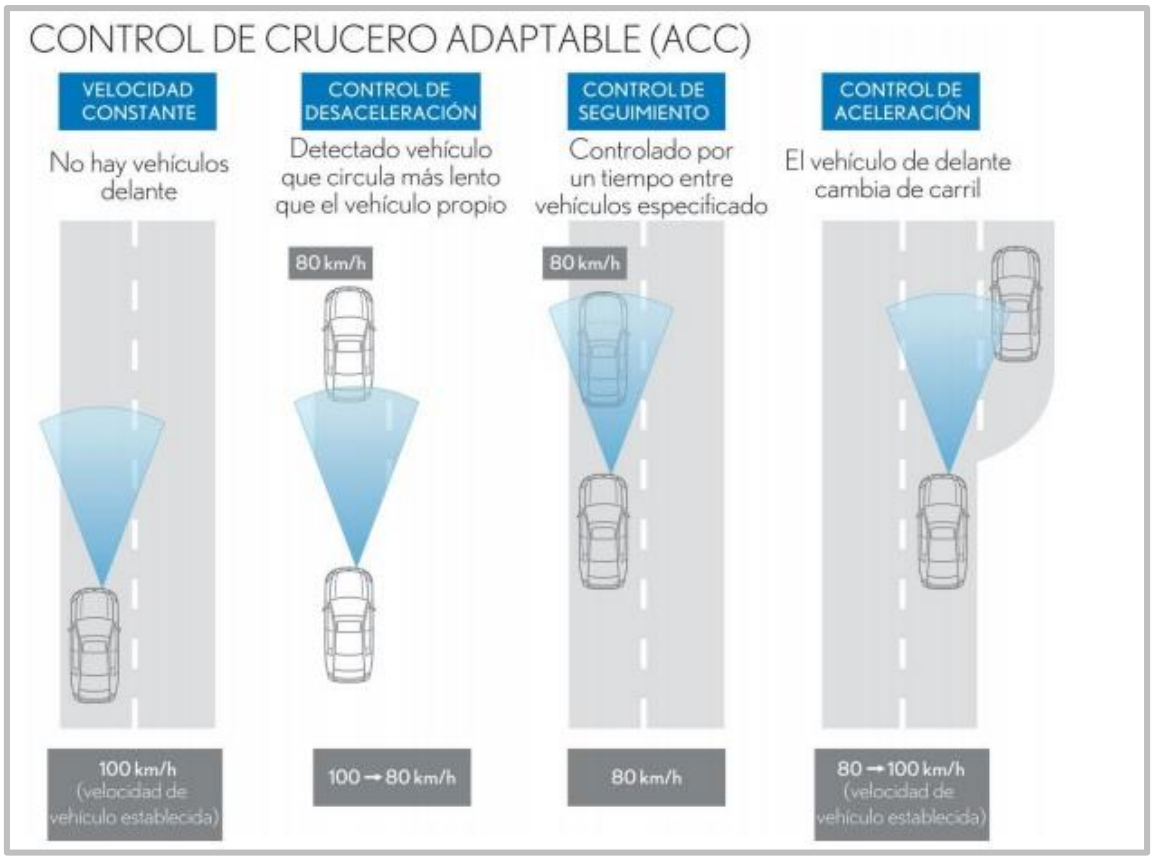


Figura 7: Funcionamiento del control de crucero adaptativo.

El funcionamiento técnico de este sistema se basa en la fusión de datos de dos dispositivos, un láser y una videocámara. El láser detecta la distancia del obstáculo con un alcance máximo de 130 metros, y se complementa con la visión por computador con una videocámara capaz de diferenciar los obstáculos a aproximadamente 50 metros, distancia variable según las condiciones meteorológicas. Con este sistema de fusión se consigue una eficiencia óptima y un rechazo a falsos obstáculos como pudieran ser coches en el carril contrario, o vehículos aparcados. También existen otros tipos de funcionamiento solamente con el dispositivo láser, pero no son tan fiables al depender del reflejo del láser, y este poder variar por ejemplo según las condiciones meteorológicas como la niebla.

La última evolución de los controles de crucero adaptativo, es el Braking Guard de Audi, donde el control avisa al conductor cuando el vehículo que le precede frena bruscamente. No hay duda que este tipo de avances ayudarán a evitar accidentes de tráfico y que salvará muchas vidas.

2.2.2. Sistema de alerta de cambio de carril (LDWS).

El sistema de alerta de cambio de carril (LDWS, L*ane departure warning system*) es un sistema que alerta al conductor si abandona el carril por el que circula sin conectar antes los intermitentes, lo que se toma por una distracción. Se compone de un sistema capaz de detectar las líneas del carril por el cual se está circulando, una centralita electrónica y un sistema de aviso al conductor.

La detección de las líneas de carril puede realizarse mediante una serie de sensores de infrarrojos instalados en la parte inferior del vehículo, que utilizan la luz reflejada por las líneas de la calzada para detectar si el vehículo circula sobre éstas. En ese caso, si el conductor no ha activado los intermitentes, una centralita electrónica interpreta que se está abandonando involuntariamente el carril y alerta al conductor mediante diversos métodos: en unos casos hace vibrar el asiento, en otros el volante y en otros emite avisos sonoros y luminosos.

Otra forma de detectar las líneas que delimitan un carril consiste en el análisis de las imágenes provenientes de una cámara (instalada generalmente en el pie del espejo retrovisor interior). Este sistema presenta como ventaja fundamental la posibilidad de reaccionar ante una trayectoria conocida, pudiendo predecir la salida de carril antes de que ésta se produzca. Además, es un sistema que no genera falsas alarmas ante otras líneas, como son flechas pintadas en la calzada. Su funcionamiento en casos de visibilidad reducida es peor.

En todo caso, el sistema funciona únicamente a partir de una cierta velocidad (60 – 80 km/h es lo más habitual) y puede ser desconectado por el conductor. Asimismo, al activar el intermitente correspondiente, se interpreta que el conductor realmente desea realizar la maniobra de abandonar el carril por el que se circula y, por lo tanto, éste no es alertado.

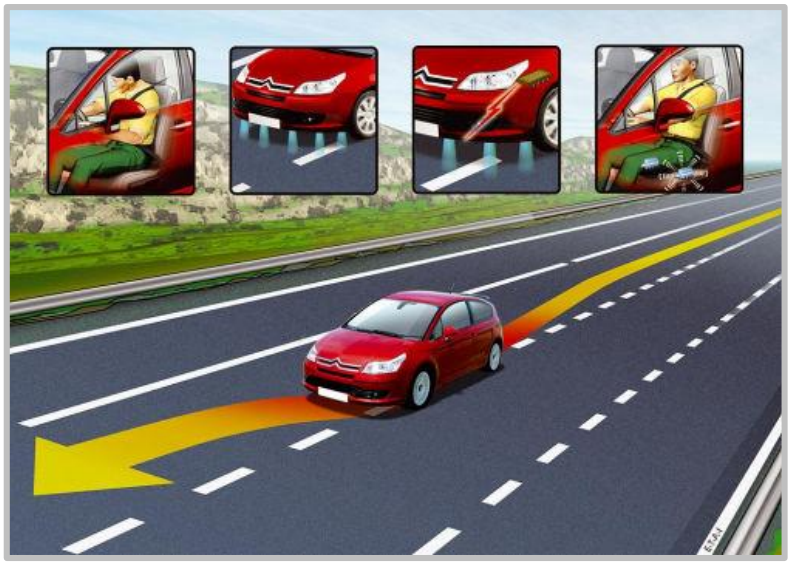


Figura 8: Funcionamiento del sistema de alerta de cambio de carril.

2.2.3. Sistema de detección de obstáculos en el ángulo muerto y asistencia para cambio de carril.

El sistema de detección de objetos en ángulo muerto tiene como objetivo alertar al conductor si hay otro vehículo en el ángulo muerto de los retrovisores o cerca de él.

Para ello, consta de sensores que vigilan constantemente la zona lateral próxima al coche. Generalmente o bien son sensores radar de corto o medio alcance a 24 GHz, o bien sistemas de procesado de imágenes. Estos sensores proporcionan información a una centralita de control que, en caso necesario, emite un aviso acústico, visual o táctil (o una combinación de varios).

Existen sistemas que pueden alertar de forma continua de la existencia de vehículos en el ángulo muerto independientemente de las intenciones del conductor, mientras que otros únicamente actúan cuando se expresa la voluntad de efectuar un cambio de carril mediante el uso del intermitente.

Con el objetivo de minimizar las falsas alarmas, generalmente actúan por encima de un umbral de velocidad determinado y son capaces de realizar un filtrado de vehículos estacionados o de aquellos que circulan en sentido contrario. La zona de detección es unos 10 metros por detrás del espejo retrovisor por unos 4 de anchura, suficiente para cubrir el ángulo muerto.

El primer vehículo en integrar el sistema de detección de objetos en ángulo muerto fue el Volvo XC90 (Otoño 2005), utilizando para ello un par de cámaras CMOS integradas en los espejos retrovisores exteriores.

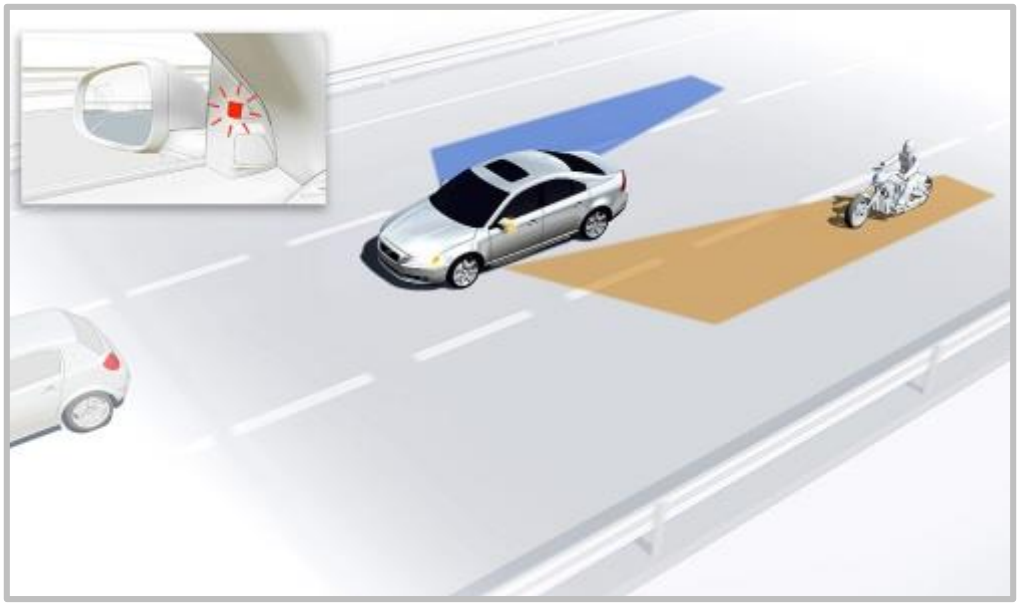


Figura 9: Funcionamiento del sistema de detección de obstáculos en el ángulo muerto.

El asistente para cambios de carril es un sistema que amplía las posibilidades de funcionamiento del sistema de detección de objetos en ángulo muerto.

La distancia de detección aumenta hasta 50 ó 60 metros por detrás del vehículo y en los carriles adyacentes al mismo. Tiene en cuenta, además, la velocidad relativa del vehículo detectado en dicha zona con respecto al propio. De esta forma, se está en disposición de alertar al conductor en caso de existir un cierto riesgo al efectuar la maniobra de cambio de carril debido a la aproximación de otro vehículo a gran velocidad. En función de diversos parámetros, se pueden establecer diversos niveles de alerta.

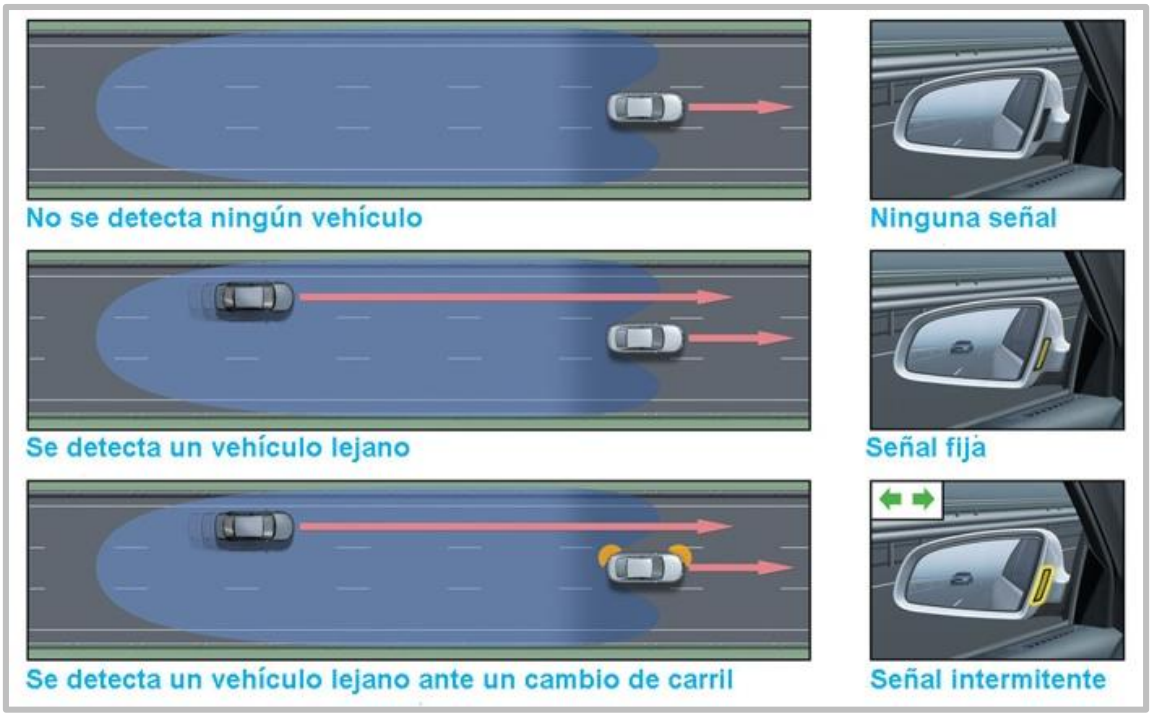


Figura 10: Funcionamiento de asistente de cambio de carril.

2.2.4. Sistema de visión nocturna.

Un sistema de visión nocturna aprovecha la emisión infrarroja, invisible al ojo humano, para captar imágenes que puede que no se perciban con la luz visible de los faros. Las imágenes que captan los sensores de luz infrarroja se ofrecen al conductor en una pantalla.

Existen dos sistema de visión nocturna basados en emisión infrarroja: los denominados pasivos o térmicos, que captan emisión en el infrarrojo lejano (FIR, Far InfraRed), y los sistemas activos, que aprovechan el infrarrojo cercano (NIR, Near InfraRed).

Los sistemas de visión nocturna pasivos aprovechan la emisión espontánea de radiación infrarroja de todos los cuerpos al estar a una cierta temperatura. Esta emisión (con una longitud de onda de entre 0.8 y 1.2 micrómetros) es captada por una cámara especial, que ha de ir situada en el exterior del vehículo, y se representa como imagen en la

pantalla situada frente al conductor. Esta imagen se muestra como un negativo, con colores más claros para los objetos a mayor temperatura, destacando así cuerpos calientes, como un peatón o un animal que cruza la carretera.

Figura 11: Sistema de visión nocturna pasivo.

Los sistemas activos se basan en un sensor de tipo CMOS sensible a la radiación infrarroja cercana. Dado que los cuerpos no emiten de forma espontánea en dicha longitud de onda (entre 780 y 1300 nanómetros), se precisa una iluminación adicional infrarroja que proporcionan dos lámparas halógenas. La imagen que se forma con esa iluminación infrarroja la recoge el sensor y la muestra una pantalla de blanco y negro.

Figura 12: Sistema de visión nocturna activo.

Un sistema pasivo tiene mayor alcance que un sistema activo (unos 300 m frente a 150 m). Además, no se ven afectados por la iluminación de otros vehículos ni por las inclemencias meteorológicas, como sí sucede con los sistemas activos, y pueden representar con mayor nitidez los objetos susceptibles de causar mayor riesgo en

conducción nocturna. Esta tecnología es todavía propiedad del ejército estadounidense, por lo que su uso está sujeto a diversas regulaciones.

Por el contrario, un sistema activo es más económico, de uso libre y tiene capacidad para detectar objetos invisibles para el sistema térmico (por ejemplo, un tronco en la calzada). Común para ambos sistemas es la posibilidad de postprocesar la imagen obtenida y realizar así funciones de reconocimiento de peatones, por ejemplo, mediante visión artificial.

2.2.5. Sistemas de detección de fatiga y somnolencia en el conductor.

Los sistemas de detección de fatiga y somnolencia en el conductor son unos sistemas preventivos que sirven para monitorizar al conductor y controlar en qué estado está conduciendo, siendo capaces de reconocer si está cansado o fatigado, o si le está entrando sueño. Los sistemas más avanzados incluso pueden reconocer si está distraído y mirando a otra parte. Viene a haber dos sistemas principalmente: por un lado se encuentran aquellos sistemas que monitorizan el entorno del vehículo y los parámetros de conducción y por otro lado aquellos sistemas que directamente monitorizan los rasgos faciales del conductor.

Los sistemas basados en el análisis de los parámetros de conducción analizan el comportamiento de conducción del conductor para crear un perfil específico de dicho comportamiento. Este tipo de sistemas utilizan sensores instalados en el volante del coche para aprender sobre nuestro manejo del volante en condiciones normales, y así detectar cuando no lo hacemos de igual forma, en cuyo caso entenderán que el conductor está distraído, fatigado o somnoliento.

La diferencia está en que cuando conducimos con normalidad hacemos pequeñas correcciones para mantenernos en el carril. Sin embargo, cuando estamos muy cansados o aparece el sueño, ya no manejamos igual el volante, y solemos hacer correcciones bruscas.

Es en ese momento cuando el sistema alerta al conductor haciendo saltar una alarma sonora y mostrando un mensaje en la pantalla del cuadro de instrumentos indicando que, por seguridad, es necesario parar a descansar.

Figura 13: Sistema de advertencia de fatiga del conductor.

El otro tipo de sistema de detección de fatiga se basa en la monitorización de los rasgos faciales del conductor, interpretando el movimiento de sus ojos y su velocidad de parpadeo como datos para la detección de la fatiga.

El funcionamiento de este tipo de dispositivos para el coche se fundamenta en el uso de una cámara, que se coloca sobre el volante, y un sistema de reconocimiento facial. Gracias a la combinación de estos elementos, la electrónica de nuestro vehículo puede conocer con precisión si sufrimos cansancio, fatiga, sueño o incluso falta de concentración, y tomar medidas al respecto para evitar un posible accidente.

La cámara enfoca a la cara del conductor y supervisa los ojos de éste para comprobar si el parpadeo es normal o si indica sueño. Asimismo, la cámara es también capaz de ver si el conductor mira al frente o si desvía la mirada fuera de la carretera, retirando su atención de la circulación.

Además de los ojos del conductor, los sistemas de reconocimiento facial son también capaces de reconocer expresiones faciales que revelan cansancio y fatiga como, por ejemplo, los bostezos. De hecho, pueden incluso reconocer si el conductor está estresado, nervioso o colérico, lo que supondría un riesgo para la conducción.

En caso de detectar cualquiera de estos indicios de fatiga o sueño, y al igual que los sistemas basados en sensores de movimiento del volante, se pondría en marcha un sistema de alarma para alertar al conductor de la situación mediante luces parpadeantes o un mensaje de texto en el cuadro de instrumentos. Aun así, que el conductor haga caso del aviso emitido es totalmente voluntario.

Los sistemas con reconocimiento facial más sofisticados están todavía en fase experimental y no pueden encontrarse en los coches de calle. No obstante, los conductores profesionales disponen ya de sistemas que funcionan, aunque aún no se trata de sistemas muy sofisticados.

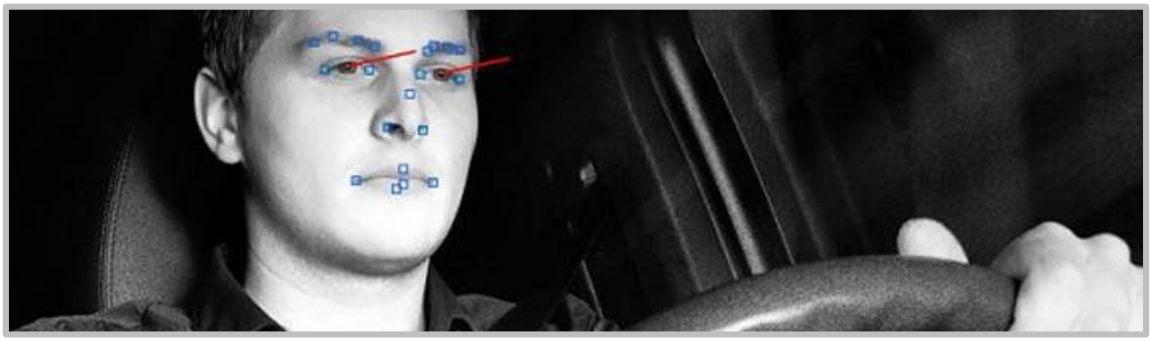


Figura 14: Funcionamiento de la cámara de reconocimiento facial.

2.2.6. Sistema de detección de señales de tráfico.

El sistema de detección de señales de tráfico se trata solamente de un sistema de advertencia, es decir, siempre queda en manos del conductor la decisión final de rebasar o no el límite de velocidad. Sin embargo, su funcionamiento es excelente y, seguramente, a no mucho tardar, se ofrecerán nuevas posibilidades de cara al futuro, en conjunción con otros sistemas electrónicos, para aumentar la seguridad de los usuarios.

Su funcionamiento se basa en una cámara que suele estar integrada en la zona media del parabrisas, junto a la base del retrovisor interior. Se suelen emplear cámaras dotadas de sensores de tipo CMOS (una tecnología muy barata y capaz de ofrecer un elevado "rango dinámico"), ya que son capaces de captar imágenes en condiciones de iluminación muy distintas.

El único inconveniente es que las imágenes no son de mucha calidad (en blanco y negro, y con una resolución de 640×480 pixel). Pero no representa ningún inconveniente porque, en cambio, su tasa de captura es relativamente alta (unos 30 fotogramas por segundo).

Las imágenes son enviadas a un cerebro electrónico que es capaz de realizar múltiples tareas de forma simultánea como, por ejemplo, detectar señales de tráfico, líneas que delimitan los carriles y otro tipo de obstáculos; también es capaz de controlar el encendido automático de las luces largas o de carretera. Cuando llega una imagen, el procesador, primero, la analiza para detectar los contornos de las señales de tráfico, después examina cada zona de la imagen encerrada por uno de esos contornos, mejorando el contraste y, al final, compara este análisis con la biblioteca de señales que tiene almacenada.

Como consecuencia el sistema es capaz de leer una señal a más de 100 metros, incluso si se trata de indicaciones mostradas en paneles electrónicos o de obras. De momento sólo son capaces de detectar los límites de velocidad, las prohibiciones de adelantamiento y las líneas de la carretera.

El límite de velocidad vigente en cada momento se muestra en la instrumentación o proyectada sobre el parabrisas (Head-Up display). La indicación se actualiza automáticamente en el momento exacto en que el coche rebasa la señal de la nueva limitación.

Figura 15: Funcionamiento del sistema de reconocimiento de señales.

2.2.7. Sistema de asistencia en intersecciones.

El objetivo de este sistemas es reducir (y a largo plazo evitar) las colisiones que se producen en las intersecciones, sobre todo enfocado a las colisiones laterales, que son las más comunes por la falta de visibilidad.

Se trata de un proyecto que todavía no ha podido ser comercializado, y aunque está muy avanzado en su desarrollo todavía siguen abiertas algunas investigaciones para que sea completamente óptimo. Hay varias marcas trabajando en su desarrollo, pero la que mejores resultados ha obtenido es una unión de varios fabricantes de vehículos (BMW, VW, PSA y RENAULT) e instituciones (INRIA, IKA, FCS, Signalbau Huber) conocido como proyecto INTERSAFE. El proyecto inicial comenzó en febrero de 2001, de 2004 a 2008 se amplió con la versión 2.0, y en la actualidad se encuentra en desarrollo la versión 3.0. Para validar la teoría y probarla en situaciones de tráfico real se instaló el sistema en un vehículo de pruebas de la marca WV Phaeton, en esta plataforma se instalaron dos escáneres láseres (integrados en la esquina izquierda y derecha delantera del vehículo), una cámara de video y un sistema de comunicación. Como ya se ha explicado en otros sistemas ADAS, la detección del entorno viario y de los obstáculos se produce por la fusión de información dada por la visión y los láser, y con los datos obtenidos se hace una evaluación dinámica de los riesgos. Esta evaluación se basa en el seguimiento y

clasificación de objetos, comunicación con la gestión del tráfico y de los propósitos del conductor. Como resultado de la dinámica de evaluación de riesgos, pueden ser identificados los posibles conflictos con otros usuarios de la carretera y la gestión de tráfico. Cuando se detecta un peligro, el conductor es avisado, o el sistema interactúa con el vehículo para evitar la colisión.

Los resultados de las fases de evaluación son bastante esperanzadores. Ya que las pruebas de ensayo han indicado que el asistente de intersección tiene una tasa de alarma correcta del 93% en escenarios de giro a la izquierda y el 100% en escenarios laterales.

2.2.8. Sistema de control de la presión de los neumáticos (TPMS).

El TPMS *(Tyre Pressure Monitoring System)* es uno de esos elementos de seguridad activa que siendo sencillos nos ahorran problemas ya que nos recuerdan la importancia de la presión del neumático. En sí, la función del sistema es avisar al conductor de una pérdida de presión de inflado en los neumáticos.

Para explicar el funcionamiento del control de presión de los neumáticos, debemos tener en cuenta que hay TPMS directo y TPMS indirecto.

En el TMPS directo, un sensor colocado en cada rueda mide la presión de inflado y transmite el dato a una centralita, que puede ofrecer el dato desglosado por cada neumático o bien un dato total, o simplemente puede avisar cuando los datos reales no cuadran con los que tiene programados. Los sensores incorporan una pequeña batería que les da autonomía para funcionar sin depender de la energía del vehículo. Estos sensores pueden medir la presión y la temperatura del neumático, además de informar al sistema empleando ondas de baja frecuencia de su posición en el neumático y del estado de su batería. Al cambiar neumáticos, rotarlos o realizar cualquier otra operación de mantenimiento suele ser necesario volver a calibrar los sensores para evitar problemas de medición.

El TPMS indirecto no emplea sensores físicos para determinar la presión de inflado de los neumáticos, sino que mide la presión de forma indirecta, a partir de la velocidad de giro de cada rueda además de otros valores que se obtienen de forma externa. El TPMS indirecto ofrece, por tanto, valores relativos, y ese es un problema inherente al sistema. No identifica más que de forma binaria que hay un problema. Además, en condiciones de baja adherencia puede dar mediciones erróneas si durante la marcha tenemos pérdidas de adherencia con el pavimento.

Figura 16: Visualización del sistema de control de la presión de los neumáticos.

2.2.9. Luces adaptativas y asistente de luces de carretera.

El control de iluminación adaptativo ofrece una mejora del campo de visión de hasta un 70% en comparación con los sistemas de faros convencionales. Especialmente en condiciones de poca luz y en las curvas, proporciona una mejor visión de largo alcance y una mayor seguridad. El sistema de luz adaptativo tiene dos funciones principales: el sistema de iluminación dinámica y el sistema de luz de giro estática.

El sistema de iluminación dinámica en curva gira los faros en la dirección de la marcha según el grado de giro del volante a una velocidad superior a 10 km/h. Unos servomotores en las unidades del faro, mueven los faros en función del ángulo del volante y la velocidad del vehículo, con el objetivo de iluminar perfectamente el trazado de la curva. Para evitar deslumbrar a los conductores que circulan de frente, este ángulo de movimiento de la luz dinámica de curvas está limitado a 15 grados. De este modo, el conductor puede advertir con antelación tanto el trazado de la curva como los posibles obstáculos, las personas o los animales que se encuentran en medio de ella. Así el conductor gana un tiempo de reacción adicional y el riesgo de accidentes disminuye sensiblemente.

La luz de giro estática se genera por medio de una lámpara halógena independiente y de un pequeño reflector. La unidad está montada detrás del reflector de la luz de posición (dependiendo del modelo se encuentra en los faros antiniebla) e ilumina el área de giro en un ángulo de aproximadamente 35 grados a varios metros de distancia.

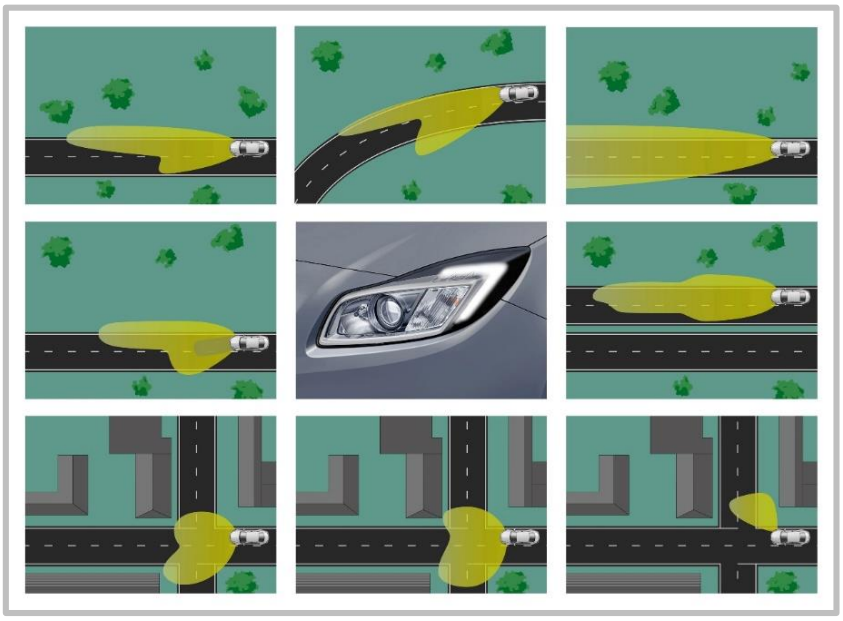


Figura 17: Funcionamiento de las luces adaptativas.

Las luces adaptativas en la actualidad suelen ir acompañadas por el asistente de luces de carretera, que puede decidir por sí mismo cuándo es conveniente utilizar las luces de carretera. Detecta de forma automática los faros de los coches que vienen de frente, las luces traseras de otros vehículos y las zonas urbanizadas, según los límites del sistema. Las luces de carretera se encienden y se apagan automáticamente dependiendo de la situación del tráfico.

Una cámara integrada discretamente en el espejo interior controla la carretera. La función de ayuda puede activarse al atardecer o por la noche, y se encarga de cambiar de forma automática entre las luces de carretera y de cruce. Junto con el sensor de luz para la activación automática de los faros, desempeña una función de ayuda de alta calidad para la iluminación.

El sistema mejora la visibilidad del conductor y proporciona una mayor comodidad al hacer de la conducción una actividad más relajante. Cuando se conduce de noche, se utilizan las luces de carretera el máximo tiempo posible, lo que hace que aumente la seguridad de forma significativa.

2.2.10. Sistema de aparcamiento automático.

Estos sistemas fueron una evolución de los primeros asistentes de estacionamiento, aquellos que incorporaban sensores de distancia en los paragolpes y emitían un pitido creciente cuando el coche se acercaba al obstáculo en cuestión.

Un sistema de estacionamiento automático requiere obligatoriamente de sensores que midan la distancia desde el coche hasta los límites de la plaza, los otros coches u obstáculos, tanto en el paragolpes trasero como en el paragolpes delantero.

Estos sensores suelen ser habitualmente de ultrasonidos y su número y distribución depende del tamaño y diseño del coche, aunque suelen ser cuatro o cinco por paragolpes. En ocasiones, aunque es poco habitual, se puede emplear un radar como sensor.

Además de estos, en los laterales del paragolpes orientados en dirección transversal, tiene que haber más sensores que midan la distancia hacia el lateral del coche. Estos sensores miden los huecos que haya para aparcar y permiten que el sistema identifique uno en el que quepa el coche.

Así cuando el conductor activa el sistema pulsando simplemente un botón, y circula junto a la banda de estacionamiento y pasa por delante de un hueco, el sistema le dirá en la pantalla del cuadro de instrumentos que ese hueco vale.

Algunos sistemas se complementan con una cámara de vídeo para visión marcha atrás, e incluso marcha adelante y 360 grados, para que el conductor supervise con mayor seguridad la maniobra.

Así que el sistema avisa al conductor de que hay un espacio válido, y le indica hasta dónde tiene que avanzar y dejar el coche. Y así le seguirá dando instrucciones en la pantalla en el momento oportuno: para insertar la marcha atrás, para acelerar, para frenar, para insertar la marcha hacia adelante, etc.

El sistema se encargará de controlar la dirección, y hará girar el volante lo que sea preciso, y cuando sea preciso, sin error, gracias a un electromotor.

Figura 18: Visualización del sistema de aparcamiento automático.

2.2.11. Sistema de comunicación entre vehículos.

La comunicación entre vehículos es un conjunto de tecnologías que permiten el intercambio de datos entre vehículos, de forma que se puedan llevar a cabo acciones orientadas a mejorar la fluidez en el tráfico, aumentar la seguridad mediante la correcta anticipación de eventos, y comunicarse de forma más eficiente con las fuerzas de seguridad y los servicios de emergencias, entre otras funciones.

Si somos más estrictos, las tecnologías de comunicación entre vehículos dan lugar a redes de comunicaciones ad hoc en las que los nodos son los propios coches. Son los que reciben información, los que intercambian datos y generan nuevas señales (o informaciones). Así, un coche (nodo) puede recoger información del tráfico en su entorno inmediato y difundirlo a través de la red de comunicación hacia otros coches: estos podrán interpretar esos datos y ofrecérselos al conductor, que tomará la decisión más adecuada.

En otros casos, los nodos servirán como una baliza de emergencia. Tras un accidente, el nodo puede difundir a la red de comunicación (en este caso sería mejor a la red de emergencias) información sobre la posición del vehículo, el estado aparente del conductor, parámetros como la fuerza del impacto o los daños registrados, y esa información puede servir a los servicios de emergencia para tomar decisiones en base a la aparente gravedad del suceso.

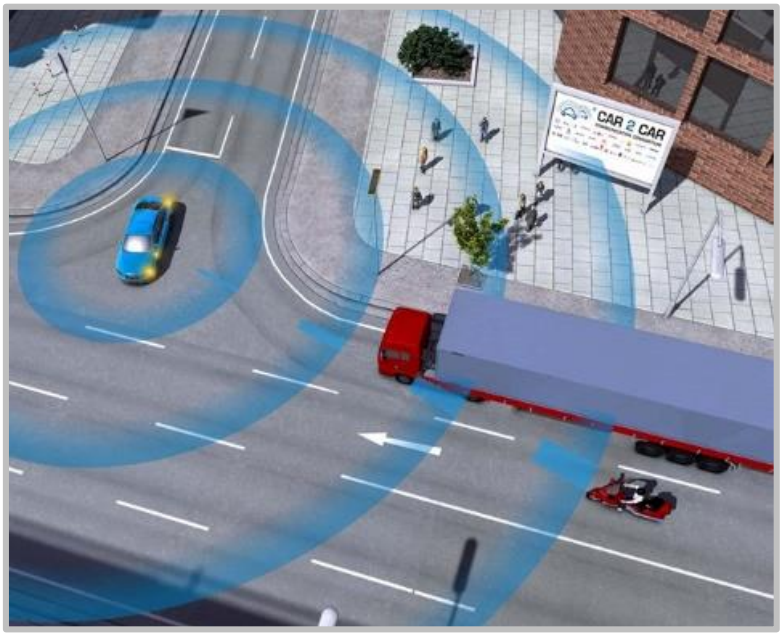


Figura 19: Ejemplo de funcionamiento del sistema de comunicación entre vehículos.

2.2.12. Sistema de llamada de emergencia (eCall).

El sistema eCall tiene como objetivo ayudar a salvar vidas, ya que avisará de manera automática a Emergencias 112 en caso de accidente. De esta manera, la velocidad de atención a los heridos será más rápida, aumentando la posibilidad de supervivencia y reducir los daños para evitar secuelas irreversibles.

A la hora de producirse un accidente de tráfico, en muchas ocasiones el tiempo de espera desde que se produce el siniestro hasta que los servicios de emergencia se hacen eco de lo ocurrido y lleguen al lugar de los hechos, puede ser lo suficientemente largo como para que no se pueda hacer nada por salvar la vida de los accidentados. Por ello, y para que la cifra de muertes sea cada vez menor, la seguridad vial intenta mejorar con la implantación en España y Europa del sistema de emergencia eCall, un dispositivo que ayuda de manera automática al auxilio del conductor y pasajeros en caso de accidente grave.

El sistema eCall detecta si un suceso es grave gracias a unos sensores instalados en el vehículo, y en ese momento se graba un mensaje de auxilio que llegará a la centralita del 112 mediante un sistema de GPS, con el que se podrá detectar fácilmente al vehículo. La llamada podrá ser realizada de manera manual por los ocupantes o automáticamente en caso de accidente grave.

Según varios estudios científicos se establece que el 70% de las muertes en accidentes de tráfico se produce a los 20-30 minutos de haberse producido, es por ello que el interés principal de este mecanismo es la instantaneidad con la que se enviará la localización del siniestro y reducirá el tiempo de respuesta de los servicios de emergencia. Según los estudios realizados en la Unión Europea, dicho tiempo de respuesta podría disminuir un 50 % en las zonas rurales y un 40 % en las zonas urbanas.

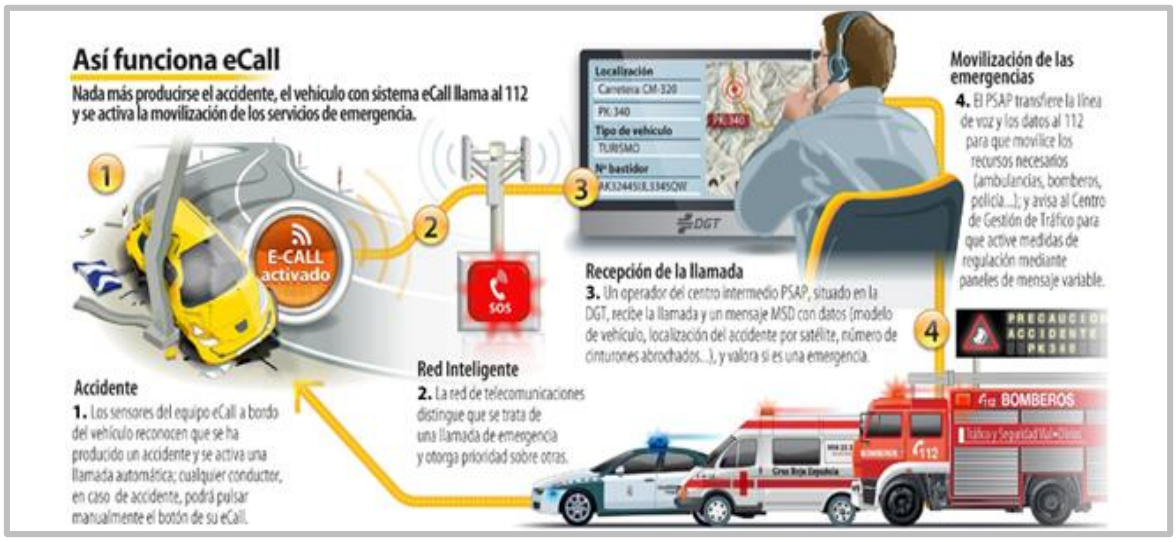


Figura 20: Funcionamiento del sistema eCall.

2.3. Vehículos autónomos.

Un vehículo autónomo es un automóvil capaz de imitar las capacidades humanas de manejo y control. Como vehículo autónomo es capaz de percibir el medio que le rodea y navegar en consecuencia.

Los vehículos autónomos perciben el entorno mediante técnicas complejas como láser, radar, lidar, sistema de posicionamiento global y visión por computador. Los sistemas avanzados de control interpretan la información para identificar la ruta apropiada, así como los obstáculos y la señalización relevante. Los vehículos autónomos generalmente son capaces de recorrer carreteras previamente programadas y requieren una reproducción cartográfica del terreno, con lo cual si una ruta no está recogida por el sistema se puede dar el caso que no pueda avanzar de forma coherente y normal.

Desde hace tiempo existen vehículos industriales útiles para el transporte de mercancía que no requiere de un conductor para su circulación, es decir, se mueve de manera automática. Estos vehículos reciben el nombre de AGV (en inglés Automatic Guided Vehicle) y mediante un sistema de guiado, ya sea por sensores o por cámaras, se garantiza el desplazamiento de materiales sin que interfiera un operario. Sin embargo, lo que se busca ahora es la adaptación de los coches con el fin de que se guíen de forma totalmente autónoma.

En el mundo hay varios programas activos, pero para su implantación definitiva se requiere de un ajuste de varios aspectos derivados de la seguridad vial y en materia de seguros. Son algunas de las dudas que concierne una forma de transporte que está cerca de ser realidad en pocos años según empresas involucradas en su desarrollo, como Google, Daimler AG, BMW, Renault, Ford o Volvo, así como Bosch o Delphi, en el área de componentes y electrónica.

2.3.1. Vehículo autónomo de Google.

El Google Driverless Car es un proyecto muy avanzado, llevado a cabo por Google, equipado con una tecnología que permite que un vehículo circule sin la necesidad de que lo controle una persona.

Estos coches han sido diseñados para evitar los errores humanos a partir de la idea de que un equipo informatizado reacciona mejor y más rápidamente que un ser humano, pues este último puede sufrir distracciones, situaciones de somnolencia o alguna otra adversidad durante la conducción. Además de aumentar la seguridad en las carreteras reduciendo el número de accidentes, estos vehículos realizan una conducción más eficiente desde el punto de vista energético.

En 2010 fue lanzado un prototipo capacitado para guiarse mediante los propios mapas de Google Maps. El automóvil recorrió el Estado de California, realizando un total de 225.000 kilómetros sin ser conducido por una persona, únicamente bajo supervisión.

El funcionamiento del coche con piloto automático de Google se basa en cámaras de vídeo, sensores de radar, rayos láser, y una base de datos de la información que va recogiendo in situ sobre los vehículos que circulan por la carretera y guiándose mediante los mapas de Google Maps. En la Figura 21 se muestran algunos de los componentes que hacen posible la conducción autónoma. Uno de los sensores se encuentra situado en el techo del vehículo y consiste en un sensor de rotación lidar con un alcance de 70 metros en todas direcciones (360 grados), proporcionando un preciso mapa tridimensional del entorno del vehículo.

El coche dispone de tres radares frontales y uno en la parte trasera encargados, también, de detectar los objetos cercanos al vehículo con una menor resolución, pero con mayor alcance. En una de las ruedas se localiza otro sensor que mide pequeños movimientos realizados por el vehículo y ayuda a localizar de forma precisa su posición en el mapa. Una cámara frontal delantera instalada cerca del retrovisor detecta las luces de los vehículos que se aproximan o se alejan y ayuda al ordenador de a bordo del coche a reconocer obstáculos en movimiento como peatones o ciclistas. La interpretación de los datos sensoriales está estrechamente ligado al problema de la localización del vehículo, que necesita conocer la localización del vehículo con una precisión mayor a la que ofrece un GPS hoy en día, por lo que para este fin se emplean sistemas que combinan GPS y RTK (Real Time Kinematic).

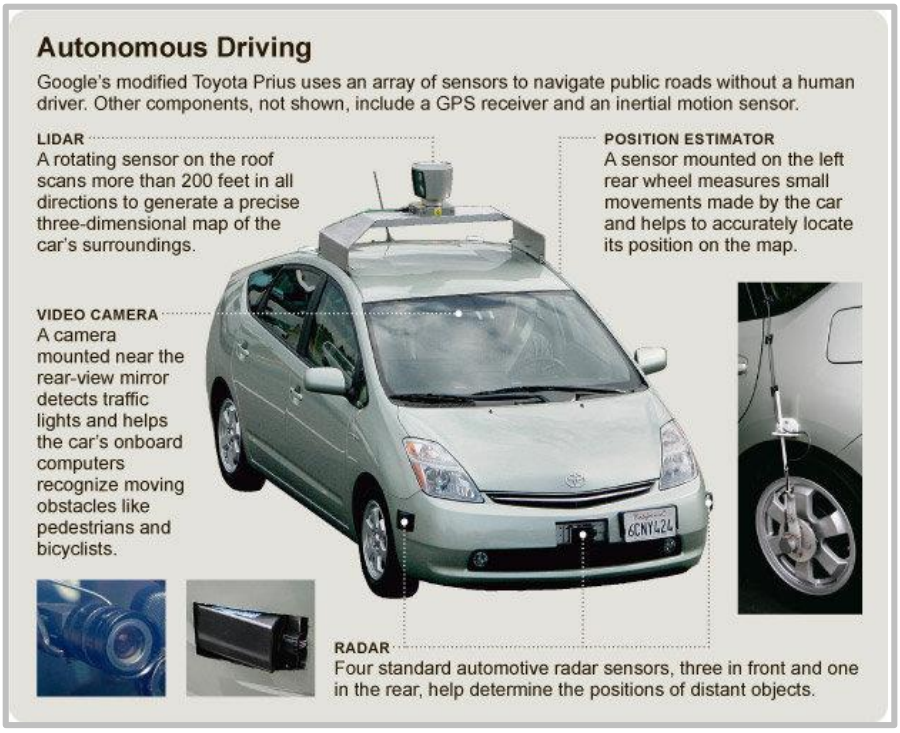


Figura 21: Componentes del Google Driverless Car.

3. DESCRIPCIÓN GENERAL.

En este apartado se va a hacer una breve descripción del diseño del algoritmo, así como una descripción del software y del hardware empleado para la realización del mismo.

3.1. Introducción.

Como se ha mencionado anteriormente, el objetivo del presente proyecto es diseñar un algoritmo que permita la detección de coches mediante un sistema de visión acoplado al vehículo.

Para ello, en primer lugar necesitamos diseñar un programa que nos vaya mostrando todas las imágenes de una secuencia obtenida previamente, de manera que podamos recortar la región que nos interesa (ROI) y almacenar las características de ese recorte en una base de datos. Tendremos que aplicar este programa tanto a secuencias de coches, que serán nuestros valores positivos, como a secuencias en las que aparezca cualquier otro tipo de obstáculo que se pueda encontrar en la vía urbana, como pueden ser peatones o motos, que corresponderán a valores negativos.

En segundo lugar, diseñaremos un programa para realizar el entrenamiento. Para ello, primero tenemos que extraer las características HOG de las imágenes. Previamente debemos redimensionar nuestras imágenes para que todas tengan el mismo tamaño y se puedan extraer las características HOG. Una vez hecho esto, se realiza el entrenamiento propiamente dicho, de manera que el programa aprende las características más relevantes en los casos positivos y en los casos negativos. Finalmente, se prueba a predecir con un cierto número de imágenes para ver si el programa ha aprendido correctamente a detectar imágenes positivas y negativas.

En último lugar, cogeremos una secuencia de imágenes, le aplicaremos el clasificador obtenido anteriormente a cada una de las imágenes y montaremos un vídeo en el que se vea la evolución de las detecciones. Para analizar los resultados se emplearán matrices de confusión.

3.2. Descripción del software empleado.

La realización del presente proyecto se ha llevado a cabo utilizando el software Qt Creatror. Esta IDE (entorno de desarrollo integrado) multiplataforma se centra en proporcionar características que ayudan a los nuevos usuarios de Qt a aprender y comenzar a desarrollar rápidamente. Qt presenta las siguientes características:

- Editor de código con soporte para C++, QML y ECMAscript.
- Herramientas para la rápida navegación del código.
- Resaltado de sintaxis y auto-completado de código.
- Control estático de código y estilo a medida que se escribe.
- Soporte para refactoring de código.
- Ayuda sensitiva al contexto.
- Plegado de código (code folding).
- Paréntesis coincidentes y modos de selección.

Qt cuenta con un depurador visual (visual debugger) para C++ que es consciente de la estructura de muchas clases de Qt, lo que aumenta la capacidad de mostrar los datos de Qt con claridad. Además Qt Creator muestra la información en bruto procedente de GDB de una manera clara y concisa.

- Interrupción de la ejecución del programa.
- Ejecución línea por línea o instrucción a instrucción.
- Puntos de interrupción (breakpoints).
- Examinar el contenido de llamadas a la pila (stack), los observadores y de la variables locales y globales.

Además, cuenta con un diseñador de interfaces gráficas de usuario (GUI) que permite diseñar rápidamente widgets y diálogos usando los mismos widgets que se usaran en su aplicación.

Para la programación del código de nuestro algoritmo hemos empleado distintas librerías, que veremos a continuación.

3.2.1. OpenCV.

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

OpenCV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados,

aprovechando además las capacidades que proveen los procesadores multi-núcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

La versión empleada para este proyecto es la 2.3.1.

3.2.2. Boost.

Boost es un conjunto de bibliotecas de software libre y revisión por pares preparadas para extender las capacidades del lenguaje de programación C++. Su licencia, de tipo BSD, permite que sea utilizada en cualquier tipo de proyectos, ya sean comerciales o no.

Su diseño e implementación permiten que sea utilizada en un amplio espectro de aplicaciones y plataformas. Abarca desde bibliotecas de propósito general hasta abstracciones del sistema operativo. Con el objetivo de alcanzar el mayor rendimiento y flexibilidad se hace un uso intensivo de plantillas. Boost ha representado una fuente de trabajo e investigación en programación genérica y metaprogramación en C++.

Varios fundadores de Boost pertenecen al Comité ISO de Estándares C++. La próxima versión estándar de C++ incorporará varias de estas bibliotecas.

Actualmente Boost está formada por más de 80 bibliotecas individuales, incluidas las bibliotecas de álgebra lineal, la generación de números pseudoaleatorios, multihilos, procesamiento de imágenes, expresiones regulares, pruebas unitarias, entre otros. La mayoría de las bibliotecas Boost están basadas en cabeceras, funciones en línea y plantillas, por lo que no tienen que ser construidas antes de su uso.

3.2.3. SQLite.

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un solo fichero estándar en la máquina host. Este diseño

simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

SQLite usa un sistema de tipos inusual. En lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales. Por ejemplo, se puede insertar un *string* en una columna de tipo entero (a pesar de que SQLite tratará en primera instancia de convertir la cadena en un entero).

Varios procesos o hilos pueden acceder a la misma base de datos sin problemas. Varios accesos de lectura pueden ser servidos en paralelo. Un acceso de escritura sólo puede ser servido si no se está sirviendo ningún otro acceso concurrentemente. En caso contrario, el acceso de escritura falla devolviendo un código de error (o puede automáticamente reintentarse hasta que expira un tiempo de expiración configurable). Esta situación de acceso concurrente podría cambiar cuando se está trabajando con tablas temporales. Sin embargo, podría producirse un interbloqueo debido al multihilo.

3.3. Vehículo de pruebas IVVI 2.0.

La plataforma de pruebas utilizada para el desarrollo del proyecto es la conocida como IVVI 2.0 (Intelligent Vehicle based on Visual Information), desarrollado por el Laboratorio de Sistemas Inteligentes de la Universidad Carlos III de Madrid. Esta plataforma consiste en un automóvil de serie de la marca Nissan, modelo Note, que se trata de un vehículo de gama media homologado para circular por carreteras abiertas al tráfico. Gracias a este vehículo de pruebas homologado se pueden probar en situaciones reales de circulación todos los sistemas que se están desarrollando en el marco de este proyecto, y poder comparar los resultados teóricos con los prácticos. Estas pruebas sobre el vehículo y en condiciones reales de tráfico son esenciales para verificar el correcto funcionamiento de los sistemas, ya que suelen ser sistemas de seguridad en los que se necesita una certeza casi absoluta de su buen funcionamiento.

Figura 22: Vehículo de pruebas IVVI 2.0.

Se encuentra equipado con una serie de sistemas que permiten reaccionar ante una serie de situaciones de peligro. Los sistemas que se distinguen en este vehículo son:

- Sistema estéreo en blanco y negro de barrido progresivo que permite capturar imágenes en movimiento, evitando los problemas inherentes al vídeo entrelazado.
- Cámara a color para la percepción de cierta información, como puede ser la ofrecida por las señales de circulación.
- Cámara de infrarrojo lejano que capta el calor desprendido por ciertos obstáculos: peatones u otros vehículos.
- Cámara orientada enfocando al conductor para evaluar el grado de atención del mismo.
- GPS que proporciona información sobre la orientación y velocidad del vehículo al resto de sistemas de percepción.
- Iluminación infrarroja que permite trabajar de noche sin molestar al resto de ocupantes del vehículo.
- Tres ordenadores situados en el maletero del vehículo para el procesamiento de la información captada por los sistemas de visión por computador.
- Sistema de visualización situado en el salpicadero del vehículo que consta de una pantalla táctil de 9 pulgadas.
- Convertidor DC/AC conectado directamente a la batería del vehículo para obtener la alimentación eléctrica necesaria para el funcionamiento de los equipos y sensores.

IVVI 2.0 es el segundo vehículo adquirido por la universidad con el equipamiento necesario para desarrollar Sistemas de Ayuda a la Conducción. Se diferencia del primer vehículo en que además de emplear equipos más modernos, presenta tanto los ordenadores como los sensores o monitores integrados en el vehículo.

4. DESARROLLO DEL ALGORITMO.

En este apartado se va a explicar detalladamente todos los procedimientos llevados a cabo para la obtención del algoritmo para detección de obstáculos.

4.1. Fundamento teórico.

A continuación se van a explicar detalladamente algunos conceptos imprescindibles para la elaboración del proyecto.

4.1.1. Histogramas de gradientes orientados (HOG).

Los Histogramas de Gradientes Orientados se basan en la idea de que objetos, como vehículos, pueden ser caracterizados mediante su apariencia. Estos descriptores obtienen la orientación del gradiente de cada píxel obteniendo de esta manera la distribución espacial del objeto. En la Figura 23 se distinguen las diferentes etapas de este descriptor.

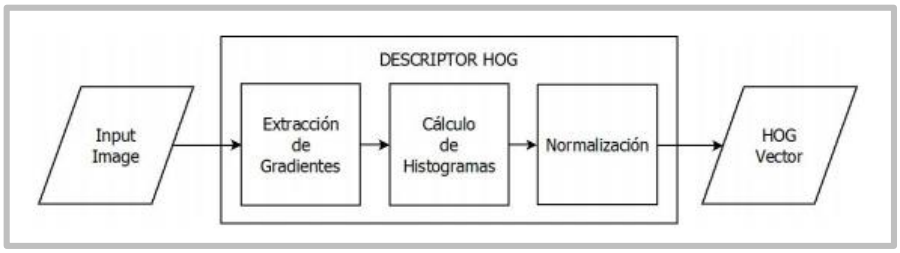


Figura 23: Sistema de obtención del descriptor HOG.

El descriptor HOG resultante se compone del cómputo de histogramas locales de orientación de los gradientes de la imagen, la cuál ha sido previamente dividida en celdas. Por lo tanto la idea principal es implementar un sistema que divida la imagen en pequeñas regiones, a las que llamaremos celdas, y obtener para cada una de ellas un histograma a partir de la orientación de los gradientes de los píxeles que la forman. Por tanto la combinación de los histogramas generados para cada una de las celdas proporcionará la representación de la imagen en el espacio de características.

Partimos, como en todos los casos, de una imagen de entrada, la cual puede ser un vehículo o cualquier otro obstáculo. En primer lugar se procede a una fase donde se van a extraer los gradientes de cada uno de los pixeles de la imagen y se obtiene a partir de estos, su módulo y ángulo. Para la extracción de gradientes podemos utilizar diferentes

métodos mediante el uso de operadores tales como: [−1, 0, 1], Sobel, Prewitt, etc. El rango de los ángulos que se utiliza es [0, π], debido a que el signo del gradiente es típicamente ignorado. Luego, se divide la imagen en celdas, donde se deben definir parámetros como el número de celdas, η1 × η2, mostrado en la Figura 24 o el número de intervalos de orientación que se va a utilizar, β.

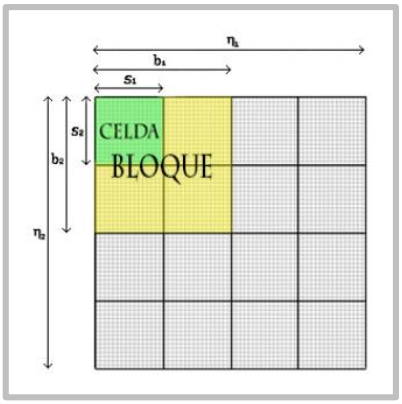


Figura 24: Ejemplo de mallado de una imagen.

En la Figura 25 encontramos un ejemplo de las diferentes configuraciones que podemos obtener cambiando el número de intervalos de orientación.

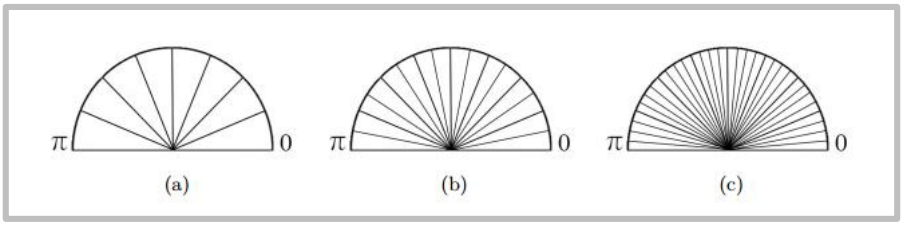


Figura 25: Posibles configuraciones cambiando el número de intervalos de orientación. El rango utilizado es [0, π] el cuál se divide uniformemente en β intervalos. (a) 8 (b) 16 (c) 32 intervalos de orientación.

Haciendo uso de estos parámetros junto al módulo y ángulo se procede al cálculo de los histogramas. La división de la imagen en celdas se puede realizar usando diferentes estructuras. Las principales topologías que se suelen seguir cuando implementamos descriptores HOG son rectangulares (R-HOG) o circulares (C-HOG).

Por último se añade una etapa de normalización que nos permite eliminar, en la medida de lo posible, los cambios de iluminación y las sombras que aparezcan en las imágenes. Para realizar esta normalización se procede a la agrupación de las celdas en estructuras de mayor tamaño que llamaremos bloques. Por lo tanto es necesario definir el tamaño de bloque, b1 × b2, un ejemplo de esto lo encontramos en la Figura 24. En muchos trabajos se propone el solapamiento de los bloques para obtener un sistema más robusto. Mediante el uso de cualquier normalización estándar se procede a la normalización de cada bloque por separado. Finalmente la concatenación de estos bloques normalizados proporciona el vector resultante HOG.

4.1.2. Máquinas de soporte vectorial (SVM).

Las Máquinas de Vectores Soporte son estructuras de aprendizaje basadas en la teoría estadística del aprendizaje. Se basan en transformar el espacio de entrada en otro de dimensión superior (infinita) en el que el problema puede ser resuelto mediante un hiperplano óptimo (de máximo margen).

Estos métodos están propiamente relacionados con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento (de muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas como pertenecientes a una u otra clase.

Más formalmente, una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta.

Se considera que es mejor clasificador de datos aquel hiperplano que maximice la distancia (o margen) con los puntos que estén más cerca de él. Siendo los vectores de soporte los puntos que tocan el límite del margen. En el contexto que se está tratando en este proyecto de detección de vehículos, las clases de datos corresponderán al vehículo (muestras positivas), mientras que el resto de obstáculos serán tachados como muestras negativas. La SVM busca un hiperplano que separe de forma óptima a los puntos de una clase de la de otra, que eventualmente han podido ser previamente proyectados a un espacio de dimensionalidad superior.

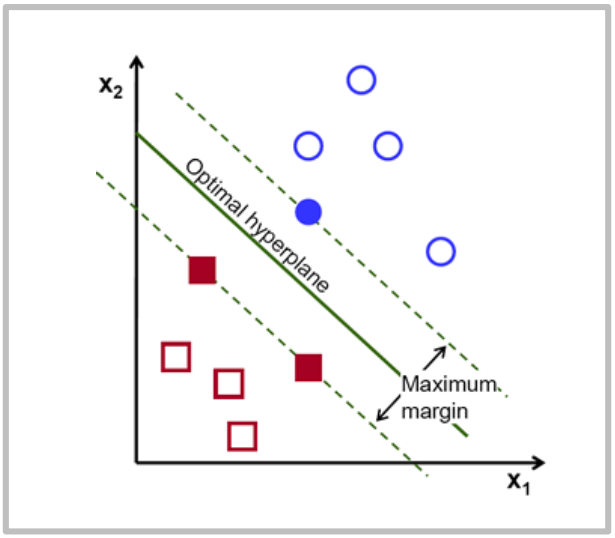


Figura 26: Hiperplano de separación de dos clases.

En ese concepto de separación óptima es donde reside la característica fundamental de las SVM. Por eso, también a veces se les conoce a las SVM como clasificadores de margen máximo. De esta forma, los puntos del vector que son etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.

Para separar linealmente los datos se procede a realizar un cambio de espacio mediante una función que transforme los datos de manera que se puedan separar linealmente. Esta función recibe el nombre de Kernel.

En este caso, los conjuntos son coches y no coches. Para ello, se necesita un entrenamiento previo de la máquina, facilitándole ejemplos de coches o "positivos" y ejemplos de no coches o "negativos". Con todos los ejemplos de entrenamiento, el algoritmo de clasificación SVM elabora una curva N-dimensional que divide ambos conjuntos, obteniendo de esta forma el kernel de la máquina. Las dimensiones del espacio dependen del número de componentes de cada vector a clasificar.

CÁLCULO DEL HIPERPLANO ÓPTIMO.

El hiperplano se denota formalmente de la siguiente manera:

$$f(x) = \beta_0 + \beta^T x,$$

donde β corresponde al vector de peso y β_0 corresponde al sesgo.

El hiperplano óptimo se puede representar de infinitas maneras diferentes ajustando β y β_0 . Entre todas las posibles representaciones de la hiperplano, el elegido es:

$$|\beta_0 + \beta^T x| = 1$$

donde x simboliza los ejemplos de entrenamiento más próximos al hiperplano. En general, los ejemplos de entrenamiento que están más cerca del hiperplano se llaman vectores de soporte. Esta representación se conoce como el hiperplano canónico.

Ahora, utilizamos el resultado de la geometría que da la distancia entre un punto X y un hiperplano (β, β_0)

$$\text{distance} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|}.$$

En particular, para el hiperplano canónico, el numerador es igual a uno y la distancia a los vectores de soporte es:

$$\text{distance}_{\text{ support vectors}} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|} = \frac{1}{\|\beta\|}.$$

Recordemos que el margen mencionado anteriormente, aquí se denota como M, es el doble de la distancia de los ejemplos más cercanos:

$$M = \frac{2}{\|\beta\|}$$

Por último, el problema de maximizar M es equivalente al problema de la minimización de una función $L(\beta)$ sujeto a algunas limitaciones. Las restricciones modelan el requisito de que el hiperplano pueda clasificar correctamente todos los ejemplos de entrenamiento x_i. Formalmente,

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}\|\beta\|^2 \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \forall i,$$

donde y_i representa cada una de las etiquetas de los ejemplos de entrenamiento.

Se trata de un problema de optimización de Lagrange que se puede resolver utilizando multiplicadores de Lagrange para obtener el vector de peso β y el sesgo β_0 del hiperplano óptimo.

4.2. Descripción del programa.

En este punto se va a explicar detalladamente los procesos llevados a cabo para la elaboración del algoritmo.

4.2.1. Etiquetador de imágenes.

El etiquetador de imágenes corresponde a la primera parte de la elaboración de nuestro programa. Como ya hemos mencionado anteriormente, el etiquetador consiste básicamente en la obtención de la región de interés de todas las imágenes de una secuencia y en su almacenamiento en una base de datos.

En primer lugar tenemos que diseñar un pequeño programa que nos permita seleccionar una región de interés de una imagen y nos guarde esa región como una nueva imagen.

Para dibujar el rectángulo empleamos la siguiente función de OpenCV:

*void rectangle(Mat& **img**, Point **pt1**, Point **pt2**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, int **shift**=0)*

Los parámetros de esta función son:

- img: es la imagen de entrada.
- pt1: es un vértice del rectángulo.
- pt2: es el vértice opuesto al pt1.
- color: es el color del rectángulo o el brillo si se trata de una imagen en escala de grises.

- thickness: es el grosor de las líneas que componen el rectángulo. Los valores negativos, como CV_FILLED, significa que la función tiene que dibujar un rectángulo relleno.
- lineType: es el tipo de línea.
- shift: es el número de bits fraccionales en las coordenadas del punto.

En la siguiente figura se muestran los distintos tipos de vista que nos interesan de nuestros obstáculos.

Figura 27: (a) vista frontal-izquierda, (b) vista frontal, (c) vista frontal-derecha, (d) vista lateral derecha, (e) vista trasera-derecha, (f) vista trasera, (g) vista trasera izquierda, (h) vista lateral izquierda.

Como ayuda a la hora de recuadrar los obstáculos, añadimos a la ventana en la que aparece dicho obstáculo una cruceta en el puntero del ratón, como se aprecia en la figura 28, de manera que nos facilitaba el trabajo a la hora de recortar y evitaba que cometiésemos errores.

Figura 28: Cruceta del ratón para facilitar el recorte.

Las regiones que nos interesan recuadrar son tanto coches, que corresponderán a nuestros valores positivos, como motos u otros tipos de obstáculos que puedan aparecer en la vía urbana, que serán nuestros valores negativos.

Figura 29: Región de interés de un coche.

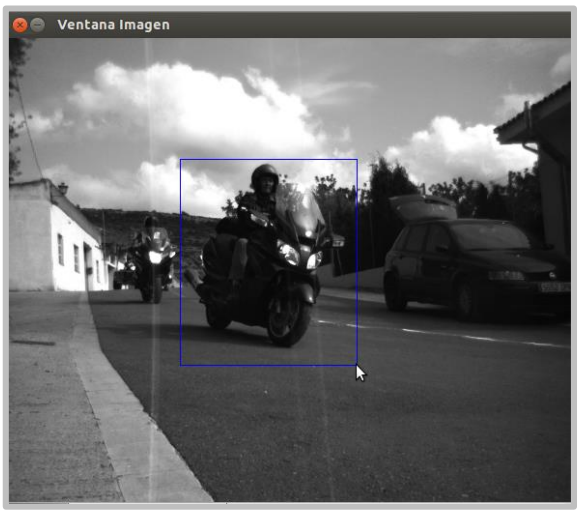


Figura 30: Región de interés de una moto.

Otro requisito importante es que a la hora de recortar el obstáculo, ya sea un coche o una moto, dicho obstáculo debe estar completo en la imagen, de manera que no se deben recortar obstáculos en los que solo se vea una parte del mismo, ya que a la hora de hacer el entrenamiento no aprendería bien las características de dicho obstáculo.

Figura 31: Ejemplo de un recorte de un obstáculo que no se ve completo.

Otra condición es que a la hora de recuadrar la región de interés el recuadro tiene que estar lo más ajustado al obstáculo como sea posible, ya que si no se ajusta al obstáculo obtendremos imágenes en las que aparezca mucho fondo, lo que puede llevar a que en el entrenamiento no se aprendan las características del obstáculo.

(a) (b)

Figura 32: (a) ejemplo de recuadro ajustado al obstáculo, (b) ejemplo de recuadro no ajustado al obstáculo.

Una vez que tenemos diseñado el programa que nos permita obtener la región de interés de una imagen, lo siguiente será adaptar ese programa para que permita obtener las ROI de todas las imágenes de un directorio. Para ello utilizamos las librerías de Boost que hemos mencionado anteriormente.

Para ello creamos un bucle con un *for* y vamos iterando en el directorio para cargar todas las imágenes.

```
for( fs::directory_iterator it( directorio ); it != end_it; it++ )
{
    cout << "Itero en directorio" << endl;
    if( fs::is_regular_file( it->status() ) && fs::extension(it->path()) == ".png"
    {
        std::cout << it->path().filename() << std::endl;

         image = cvLoadImage( it->path().string().data() );

         if ( image.empty() )
         {
              cout << "Error al cargar la imagen" << endl;
              return -1;
         }
```

Figura 33: Código del bucle para iterar en un directorio.

Una vez que tenemos el programa que muestre todas las imágenes de un directorio y nos permita seleccionar una región de interés de cada una de ellas se realiza la base de datos. Para ello, se emplean las librerías de SQlite, mencionadas anteriormente.

Para ello primero se crea la base de datos con el nombre que se quiera y se añade la base de datos a la lista de conexiones de base de datos utilizando el driver y el nombre de la conexión. Si ya existe una conexión de base de datos llamada igual, la conexión se elimina. Por ello, es conveniente comprobar previamente si se ha conectado correctamente a la base de datos.

```
QString nombre;
nombre.append("BaseDatosMotos.sqlite");

db = QSqlDatabase::addDatabase("QSQLITE");
db.setDatabaseName(nombre);

if( db.open()){
    qDebug()<<"Se ha conectado a la base de datos correctamente";
    qDebug()<<"";
}
else {
    qDebug()<<"ERROR! No se ha conectado a la base de datos";
     qDebug()<<"";

}
```

Figura 34: Código para la creación y la conexión de una base de datos.

Una vez creada la base de datos, el siguiente paso es crear una tabla dentro de la base de datos. En esta tabla es dónde se va a escribir todos los datos que queramos almacenar en la base de datos. En nuestro caso, los datos que nos interesan almacenar son: Nombre del archivo original, nombre del archivo recortado, tipo de obstáculo, tipo de vista, anchura y altura del recorte, posición X y posición Y de nuestro recorte, directorio donde se encuentra la imagen original y directorio donde se almacena la imagen recortada.

Al crear la tabla, tenemos que indicar de qué tipo es el elemento que vamos a introducir en la base de datos (int, string, float…) y la longitud máxima en caso de ser una cadena de caracteres.

Al igual que al crear la base de datos, es conveniente comprobar si se ha creado correctamente la tabla.

```
void crearTablaDetecciones(){
    QSqlQuery query;
    query.prepare("create table detecciones("
                  "original varchar(100),"
                  "crop varchar(100),"
                  "obstacle varchar(100),"
                  "view varchar(100),"
                  "width int,"
                  "height int,"
                  "xSup int,"
                  "ySup int,"
                  "originalDir varchar(100),"
                  "cropDir varchar(100)"
                  ");");

    if(query.exec()){

        qDebug()<<"La tabla DETECCIONES  se ha creado correctamente.";
        qDebug()<<"";

    }else{

        qDebug()<<"La tabla DETECCIONES ya existe o no se ha creado correctamente";
        qDebug()<<"";
    }

}
```

Figura 35: Código para la creación de una tabla dentro de una base de datos.

Después de crear la tabla, lo siguiente será crear una función para introducir datos en la tabla. Nos interesa almacenar datos tanto de la imagen original como de la imagen invertida, que creamos para obtener un mayor número de imágenes. Para obtener esta imagen invertida se utiliza la función de OpenCV:

*void flip(InputArray **src**, OutputArray **dst**, int flip**Code**)*

Los parámetros de esta función son:

- src: es el array de entrada, en nuestro caso una imagen.
- dst: es el array de salida, que tiene las mismas dimensiones que el src.
- flipCode: es un parámetro que indica como voltear la matriz. El valor 0 significa voltear la matriz alrededor del eje X. El valor 1 significa voltear la matriz alrededor del eje Y.

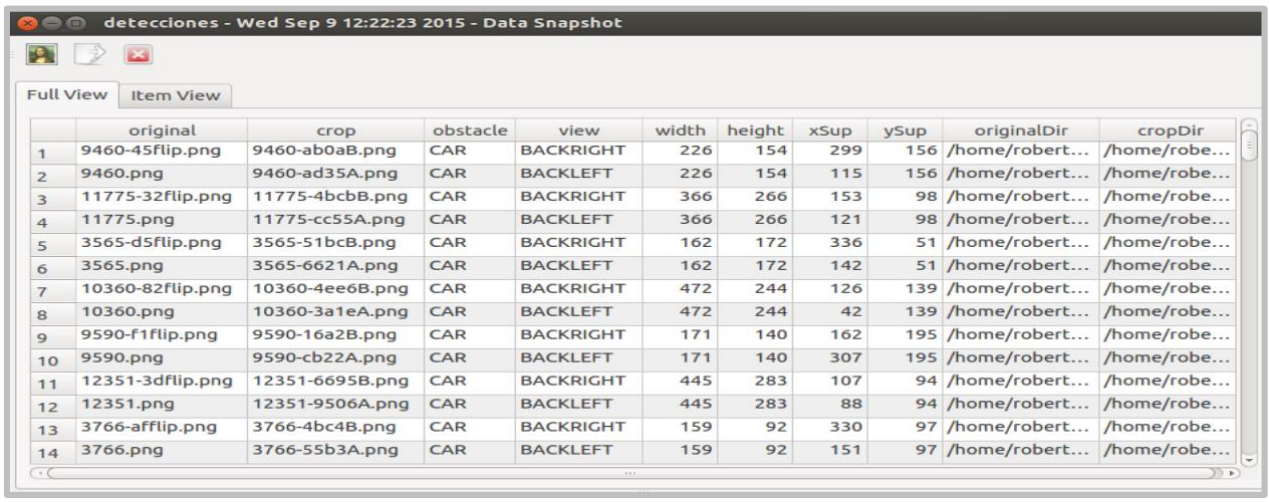
(a) (b)

Figura 36: (a) ejemplo de imagen normal, (b) ejemplo de imagen invertida.

Para guardar los datos es necesario crear una función para insertar las detecciones normales y una función para cada tipo de vista de cada imagen invertida.

Para visualizar los datos introducidos en la tabla de la base de datos se utilizó el software S*qliteman*.

En la siguiente imagen podemos observar cómo se visualizan los datos con este software.



	original	crop	obstacle	view	width	height	xSup	ySup	originalDir	cropDir
1	9460-45flip.png	9460-ab0aB.png	CAR	BACKRIGHT	226	154	299	156	/home/robert...	/home/robe...
2	9460.png	9460-ad35A.png	CAR	BACKLEFT	226	154	115	156	/home/robert...	/home/robe...
3	11775-32flip.png	11775-4bcbB.png	CAR	BACKRIGHT	366	266	153	98	/home/robert...	/home/robe...
4	11775.png	11775-cc55A.png	CAR	BACKLEFT	366	266	121	98	/home/robert...	/home/robe...
5	3565-d5flip.png	3565-51bcB.png	CAR	BACKRIGHT	162	172	336	51	/home/robert...	/home/robe...
6	3565.png	3565-6621A.png	CAR	BACKLEFT	162	172	142	51	/home/robert...	/home/robe...
7	10360-82flip.png	10360-4ee6B.png	CAR	BACKRIGHT	472	244	126	139	/home/robert...	/home/robe...
8	10360.png	10360-3a1eA.png	CAR	BACKLEFT	472	244	42	139	/home/robert...	/home/robe...
9	9590-f1flip.png	9590-16a2B.png	CAR	BACKRIGHT	171	140	162	195	/home/robert...	/home/robe...
10	9590.png	9590-cb22A.png	CAR	BACKLEFT	171	140	307	195	/home/robert...	/home/robe...
11	12351-3dflip.png	12351-6695B.png	CAR	BACKRIGHT	445	283	107	94	/home/robert...	/home/robe...
12	12351.png	12351-9506A.png	CAR	BACKLEFT	445	283	88	94	/home/robert...	/home/robe...
13	3766-afflip.png	3766-4bc4B.png	CAR	BACKRIGHT	159	92	330	97	/home/robert...	/home/robe...
14	3766.png	3766-55b3A.png	CAR	BACKLEFT	159	92	151	97	/home/robert...	/home/robe...

Figura 37: Visualización de una base de datos.

Una vez diseñado el programa para recuadrar las imágenes de un directorio y almacenar sus datos en una base de datos, se realizaron dos alternativas para la parte de la ejecución del programa.

En primer lugar se realizó un programa que permitía introducir los parámetros de las imágenes en la base de datos de forma manual a través del teclado, mediante la consola del compilador.

Como ventaja cabe destacar que de esta forma las imágenes de un directorio no tenían por qué estar ordenadas por vistas, ya que a la hora de introducir los datos se podía elegir el tipo de obstáculo y el tipo de vista. Además permitía la opción de seleccionar varias regiones de interés de la misma imagen.

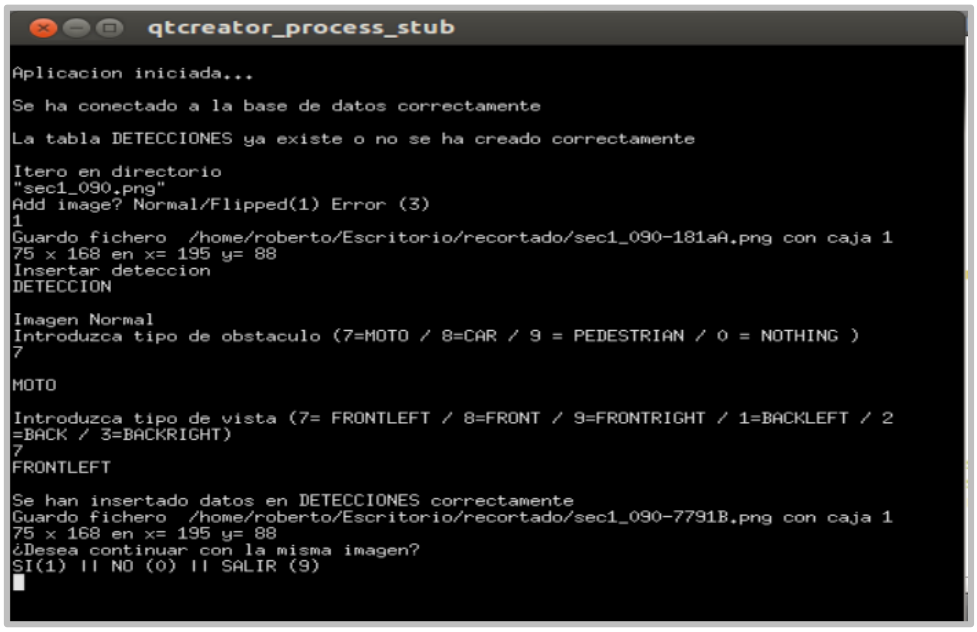


Figura 38: Ejemplo de ejecución para la introducción de datos.

Otra ventaja importante es que admitía cabida a errores a la hora de realizar el recuadro, ya que justo después de realizar el recuadro el programa te preguntaba si querías introducir los valores en una base de datos o, por el contrario, si se había cometido algún error a la hora de hacer el recuadro. De esta forma, en caso de cometer algún error, se volvía a cargar la misma imagen de nuevo y volvías a tener la opción de realizar el recuadro correctamente.

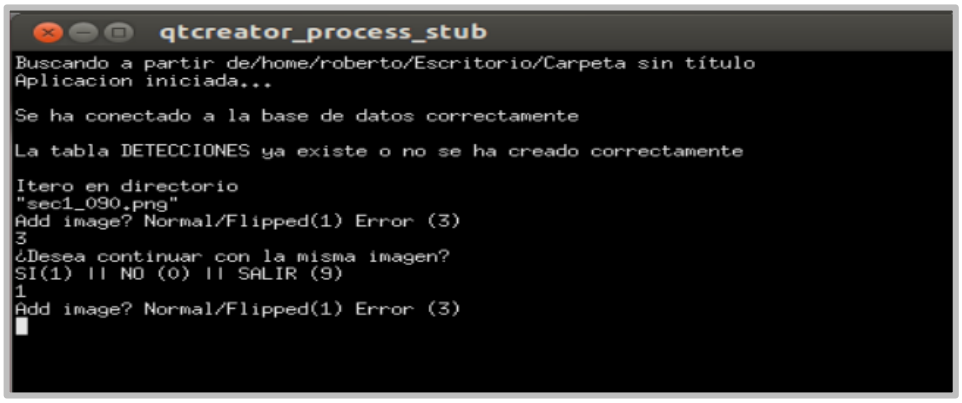


Figura 39: Ejemplo de ejecución en caso de error a la hora de seleccionar la ROI

De esta forma también se podía no realizar nada en la imagen en el caso de no haber ningún obstáculo que nos interesase recuadrar.

El principal inconveniente de este método es que es un poco lento de ejecutar, ya que cada vez que recuadramos una imagen tenemos que estar cambiando del ratón al teclado e introducir manualmente los valores que se almacenan en la base de datos.

Como alternativa a este programa, se diseñó otro programa en el que los parámetros de las imágenes se introducían directamente en la base de datos a la hora de hacer el recuadro.

Para ello, era necesario que las imágenes de un directorio estuviesen completamente ordenadas por vistas, de manera que un directorio no podría haber imágenes de coches vistos por detrás e imágenes de coches vistos lateralmente. De esta manera, era necesaria una clasificación previa de las imágenes por vistas.

Antes de ejecutar el programa, se escribía directamente en el código del programa en la parte de la base de datos de qué obstáculo se trataba y el tipo de vista del directorio que se iba a ejecutar.

De esta forma, cuando se lanzaba el programa aparecían todas las imágenes con el mismo tipo de vista, de manera que solo se tenía que recuadrar el obstáculo y automáticamente pasaba a otra imagen.

Como principal ventaja de este método cabe destacar la rapidez en la ejecución, ya que solo se utilizaba el ratón.

El principal inconveniente es que no había cabida a fallos, ya que al hacer el recorte de una imagen se pasaba directamente a la siguiente imagen. Además, solo se podía hacer un recuadro por imagen de manera que si aparecían varios obstáculos no podíamos cogerlos todos.

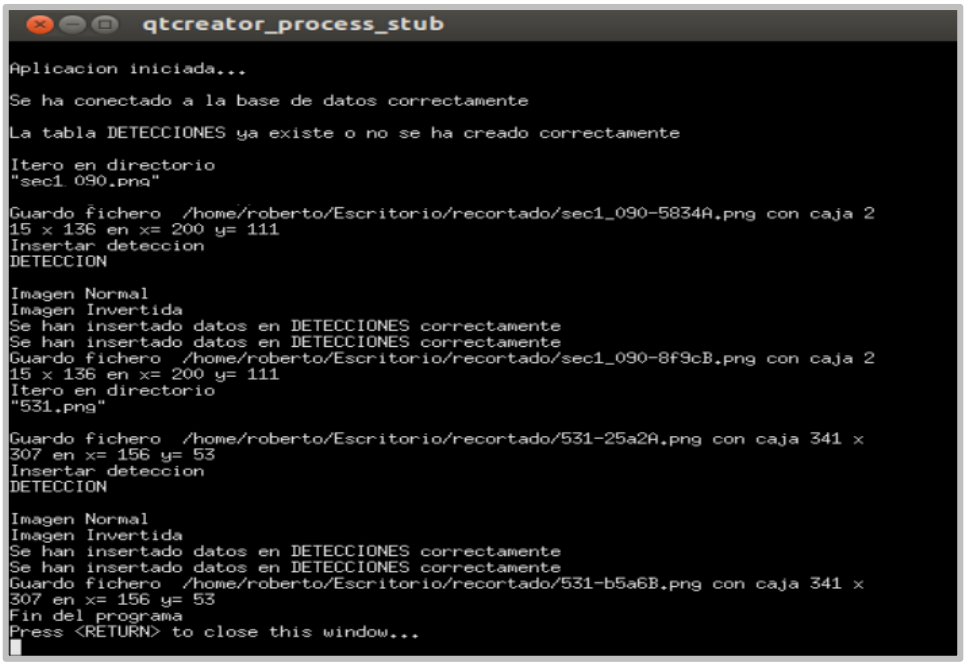


Figura 40: Ejemplo de ejecución en el caso de sólo utilizar el ratón.

4.2.2. Entrenamiento SVM.

En este punto se va a detallar como se ha llevado a cabo la parte del entrenamiento de nuestro sistema.

Para poder entrenar este clasificador se requiere un conjunto de ejemplos positivos y otro negativo.

En primer lugar es necesario tener todas las imágenes con el mismo tamaño. En nuestro caso, al querer detectar coches, utilizamos imágenes de 128x64 píxeles.

Para ello, utilizamos dos métodos diferentes.

El primer método empleado consiste en una función de las OpenCV:

*void resize(InputArray **src**, OutputArray **dst**, Size **dsize**, double **fx**=0, double **fy**=0, int **interpolation**=INTER_LINEAR)*

cuyos parámetros son:

- src: Corresponde a la imagen de entrada
- dst: es la imagen de salida, ya redimensionada.
- dsize: es el tamaño de la imagen redimensionada.
- fx: es un factor de escala a lo largo del eje horizontal.
- fy: es un factor de escala a lo largo del eje vertical.
- interpolation: es el método de interpolación.

Es un método rápido, ya que permite obtener un gran número de imágenes redimensionadas en un corto periodo de tiempo.

El principal problema de esta función reside en que realiza la redimensión a partir de nuestras imágenes recortadas, en lugar de realizarla a partir de las imágenes originales. Esto se traduce a que en algunas ocasiones aparezcan imágenes un poco distorsionadas una vez redimensionadas, como se puede apreciar en la figura 44.

(a) (b)

Figura 44: Ejemplos de imágenes redimensionadas. (a) Imagen correcta, (b) imagen distorsionada.

El otro método para redimensionar las imágenes consiste en un script de Python, mediante el cual se leen los valores de las bases de datos de nuestras imágenes, de manera que cuando se ejecuta el script se realiza la redimensión de las imágenes a partir de la imagen original, en lugar de a partir de los recortes, como se hacía mediante la función de OpenCV.

Esto se traduce en que las imágenes que se van a obtener van a salir sin distorsionar en la mayoría de los casos.

El principal problema reside en que en algún caso, como el programa toma valores de la posición del cuadrado del recorte y de las dimensiones de dicho recorte sobre la imagen original, puede que aparezca alguna imagen en la que no se vea el coche completo, por lo que tendremos que eliminarla de nuestro conjunto de imágenes.

(a) (b)

Figura 41: Imágenes redimensionadas mediante el script de Python. (a) Imagen correcta. (b) Imagen de un coche que no está completo.

Una vez redimensionadas las imágenes, tenemos que dividirlas. Utilizaremos un 70% de las imágenes para realizar el entrenamiento y obtener el clasificador, un 20% para realizar un refino, mediante técnicas de aprendizaje activo y bootstrapping, y un 10% para probar las prestaciones del clasificador. Esta división se hace tanto para las imágenes positivas como para las negativas.

Lo siguiente será extraer las características HOG de cada una de ellas.

En primer lugar se define el tamaño de las características HOG, que tiene que coincidir con el tamaño que tengan las imágenes de las que se extraen (positivas y negativas). En este caso, al tratarse de detección de coches, el tamaño será de 128x64 píxeles.

```
HOGDescriptor hog;
hog.winSize = Size(128, 64);
```

Figura 42: Definición del tamaño de las HOG.

Posteriormente se calculan las características HOG de cada imagen. Con esto se hace un fichero que es lo que luego se pasará al entrenamiento.

```
static void calculateFeaturesFromInput(const string& imageFilename, vector<float>& featureVector, HOGDescriptor& hog)

{


std::cout << "Calculating features from " << imageFilename << std::endl;

    Mat imageData = imread(imageFilename, 0);
    if (imageData.empty()) {
        featureVector.clear();
        printf("Error: HOG image '%s' is empty, features calculation skipped!\n", imageFilename.c_str());
        return;
       }

    // Check for mismatching dimensions

    if (imageData.cols != hog.winSize.width || imageData.rows != hog.winSize.height) {
        featureVector.clear();
        printf("Error: Image '%s' dimensions (%u x %u) do not match HOG window size (%u x %u)!\n",
        imageFilename.c_str(), imageData.cols, imageData.rows, hog.winSize.width, hog.winSize.height);
        return;
    }
    vector<Point> locations;
    hog.compute(imageData, featureVector, winStride, trainingPadding, locations);
    imageData.release(); // Release the image again after features are extracted
}
```

Figura 43: Función para calcular las características HOG.

A continuación se crea un fichero, que corresponde al fichero de características (features.dat), el cual tiene muchas líneas (una por imagen). Estas líneas empiezan por +1 o por -1 dependiendo de si describen una imagen positiva (con coche) o negativa (sin coche). A continuación, van numeradas cada una de las 3780 características que calculamos en el paso anterior mediante la función *calculateFeaturesFromInput.* Estas características que van numeradas tienen cada una un valor float. El conjunto de estas 3780 características de cada línea define la imagen de la que se ha extraído esa línea.

Una vez que hemos extraído las características HOG de todos los ficheros, positivos y negativos, ya sabemos cómo es el conjunto de entrenamiento.

El siguiente paso es realizar el entrenamiento.

Para ello, previamente se definen una serie de parámetros.

```
CvSVMParams params;
params.svm_type    = CvSVM::C_SVC;
params.kernel_type = CvSVM::LINEAR;
params.term_crit   = cvTermCriteria(CV_TERMCRIT_ITER, 100, 1e-6);
```

Figura 44: Definición de los parámetros de entrenamiento.

El primer parámetro corresponde al tipo de SVM. El tipo CvSVM::C_SVC se ocupa de la separación imperfecta de las clases (es decir, cuando los datos de entrenamiento no son linealmente separable). Esta característica no es importante aquí, ya que los datos son linealmente separables y elegimos este tipo SVM sólo por ser el más utilizado.

El segundo parámetro es el tipo de kernel. Es un mapeo realizado con los datos de entrenamiento para mejorar su parecido con un conjunto de datos linealmente separables. Este mapeo consiste en aumentar la dimensionalidad de los datos y se realiza de manera eficiente utilizando una función kernel. Elegimos aquí el tipo CvSVM::LINEAR, que significa que no se hace ninguna asignación.

Por último tenemos el criterio de terminación del algoritmo. El procedimiento de formación SVM se implementa resolviendo un problema de optimización cuadrática restringido de forma iterativa. Aquí especificamos un número máximo de iteraciones y un error de tolerancia por lo que permitimos que el algoritmo pueda terminar en menos pasos, incluso si el hiperplano óptimo no se ha calculado todavía. Este parámetro se define en una estructura cvTermCriteria.

Una vez definidos los parámetros, el siguiente paso es entrenar.

```
Mat labelsMat(overallSamples,1, CV_32FC1, labels);
Mat trainingDataMat(overallSamples, featuresNr, CV_32FC1, trainingData);

for (size_t i=0; i!= overallSamples; i++)
{
     std::cout<< i << " es " << labelsMat.at<float>(i) << std::endl;

    for (size_t j=0;j!=featuresNr;j++)
     {
        std::cout << trainingDataMat.at<float>(i,j) << " " ;
     }
    std::cout<< std::endl;
}


// Train the SVM
CvSVM SVM;
SVM.train(trainingDataMat, labelsMat, Mat(), Mat(), params);


 std::cout<< " \n TRAINED!!! \n"<< std::endl;

 SVM.save(descriptorVectorFile.data());
```

Figura 45: Código para realizar el entrenamiento.

Al ejecutar train se coge la información que se ha extraído del conjunto de entrenamiento y aprende de él cuáles son las características comunes a los casos positivos y cuáles a los negativos, para luego poder predecir. Después de entrenar, guardamos toda la información del entrenamiento en un fichero (*descriptorVector*), que es el clasificador, que es lo que se usa para decir si una imagen es positiva o negativa.

Este entrenamiento solo se realiza una vez, ya que tiene un tiempo de ejecución elevado.

Una vez realizado el entrenamiento, se probará el clasificador con el 20% de las imágenes de refino y en función de los resultados se aplicaran técnicas de aprendizaje activo y de bootstrapping.

```
fstream File2;
File2.open(PredictionFile.c_str(), ios::out);
if (File2.good() && File2.is_open()) {
     for (size_t i=0; i != overallSamples ; i++)
        {
               std::cout << i << " es " << labels[i] << " y predigo " << SVM.predict(trainingDataMat.row(i)) << std::endl;
             File2 << i << " es " << labels[i] << " y predigo " <<   SVM.predict(trainingDataMat.row(i)) << std::endl;
            }


        File2<<endl;
        File2.flush();
        File2.close();

        }else {
            printf("Error opening file '%s'!\n", PredictionFile.c_str());
            return EXIT_FAILURE;
        }
```

Figura 46: Código para realizar la predicción.

En la figura 46 se muestra el código mediante el cual realizaremos la predicción y, de esta forma, poder aplicar las técnicas de refino a nuestro clasificador y comprobar su funcionamiento mediante un test final.

El bootstrapping consiste en aplicar el clasificador a las imágenes negativas del 20% de refino. Si en alguna de ellas aparece un falso positivo se recorta y se añade al conjunto de "negativos" utilizados en el entrenamiento. Cuando ya tenemos todos los falsos positivos añadidos a "negativos", volvemos a entrenar. Volvemos a probar e iteramos hasta que no salgan falsos positivos.

El aprendizaje activo consiste en aplicar el clasificador al 20% de imágenes positivas de refino que hemos guardado. Si nos detecta alguna imagen como negativo, la añadimos a "positivos" y de esta manera eliminamos todos los falsos negativos. Reentrenamos y volvemos a probar, hasta que no aparezcan falsos negativos.

Finalmente, una vez aplicado las técnicas de refino al clasificador, realizamos una predicción final con el 10% de nuestras imágenes.

Para comprobar el funcionamiento del programa, cargaremos una secuencia de imágenes y comprobaremos si el programa recuadra automáticamente los obstáculos que detecte como coches. Para ello en primer lugar se volverá a cargar el clasificador. Se emplearán dos funciones para detectar el coche y realizar el recuadro automáticamente.

```
static void detectTest(const HOGDescriptor& hog, Mat& imageData) {
    vector<Rect> found;
    int groupThreshold = 2;
    Size padding(Size(32, 32));
    Size winStride(Size(8, 8));
    double hitThreshold = 2; // tolerance
    hog.detectMultiScale(imageData, found, hitThreshold, winStride, padding, 1.05, groupThreshold);
    cout<<found.size()<<endl;
    showDetections(found, imageData);
```

Figura 47: Función para recuadrar el obstáculo.

```
static void showDetections(const vector<Rect>& found, Mat& imageData) {
    vector<Rect> found_filtered;
    size_t i, j;

    for (i = 0; i < found.size(); ++i) {
        Rect r = found[i];
        for (j = 0; j < found.size(); ++j)
            if (j != i && (r & found[j]) == r)
                break;
         if (j == found.size())
            found_filtered.push_back(r);
    }
    for (i = 0; i < found_filtered.size(); i++) {
        Rect r = found_filtered[i];
        rectangle(imageData, r.tl(), r.br(), Scalar(64, 255, 64), 3);
    }

}
```

Figura 48: Función para mostrar el recuadro en la imagen.

En la figura 48 se puede observar un parámetro importante a la hora de la detección de obstáculos, que ese "*hitThreshold*". Este parámetro representa el grado de sensibilidad a la hora de detectar coches, de manera que si se utiliza un valor bajo se detectarán solo coches que se vean con claridad y cuyas características estén bien definidas en el clasificador. En cambio, si se emplea un valor más elevado, estaremos reduciendo esa restricción a la hora de detectar coches, de forma que se podrá detectar coches con una mayor facilidad pero también es posible que aparezca algún falso positivo en la imagen.

5. RESULTADOS.

En este capítulo se analizan los resultados obtenidos por el algoritmo desarrollado en el presente proyecto.

Para analizar los resultados del clasificador de una manera más visual se emplearán matrices de confusión y, a partir de ellas, se obtendrán diferentes parámetros que nos indicaran la calidad del clasificador.

5.1. Fundamento teórico.

Una matriz de confusión es una herramienta de visualización que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases.

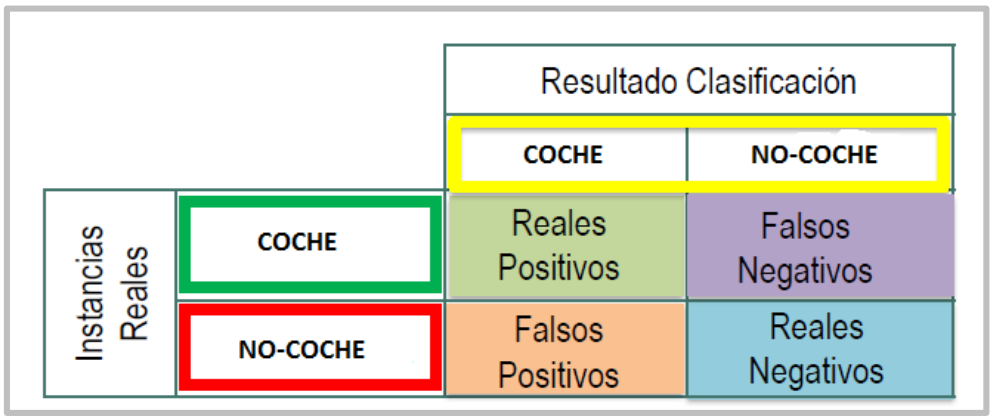


Figura 49: Ejemplo de matriz de confusión.

Mediante los valores mostrados en la matriz de confusión se pueden obtener cuatro medidas diferentes: la exactitud, la precisión, la sensibilidad y la especificidad.

La exactitud consiste en la proximidad entre el resultado y la clasificación exacta.

$$\text{Exactitud} = \frac{\text{Reales Positivos} + \text{Reales Negativos}}{\text{Predicciones Totales}}$$

Figura 50: Fórmula para el cálculo de la exactitud.

La precisión es la calidad de la respuesta del clasificador.

$$\text{Precisión} = \frac{\text{Reales Positivos}}{\text{Reales Positivos} + \text{Falsos Positivos}}$$

Figura 51: Fórmula para el cálculo de la precisión.

La sensibilidad es la eficiencia en la clasificación de todos los elementos que son de la clase.

$$Sensibilidad = \frac{\#Reales\,Positivos}{\#Reales\,Positivos + \#FalsosNegativos}$$

Figura 52: Fórmula para el cálculo de la sensibilidad.

Por último, la especificidad es la eficiencia en la clasificación de todos los elementos que no son de la clase.

$$Especificidad = \frac{\#Reales\,Negativos}{\#Reales\,Negativos + \#FalsosPositivos}$$

Figura 53: Fórmula para el cálculo de la sensibilidad.

Cabe destacar otros dos parámetros importantes a la hora de analizar los resultados:

True Positive Rate: es la tasa de reales positivos, que corresponde con la sensibilidad.

False Positive Rate: es la tasa de falsos positivos, que corresponde a (1 – Especificidad).

5.2. Análisis de las pruebas y de los resultados.

En este apartado se hablará de las distintas pruebas realizadas así como de los resultados obtenidos.

Como se ha visto anteriormente, en primer lugar realizamos un entrenamiento para obtener el clasificador. Se realizaran cuatro pruebas diferentes, dos de ellas tendrán las imágenes redimensionadas mediante la función de OpenCV vista anteriormente y las otras dos tendrán imágenes redimensionadas mediante el script de Python.

PRUEBA 1:

El clasificador se obtiene a partir del conjunto de imágenes de entrenamiento, que cuenta con un total de 5599 imágenes, de las cuales 3352 corresponden a positivos y 2247 a negativos.

La primera prueba se realizará con el conjunto de las imágenes de test que corresponden aproximadamente a un 10% del total. Se realizará la prueba sin aplicar las técnicas de bootstrapping ni de aprendizaje activo. Estas imágenes, junto con las imágenes de entrenamiento, han sido redimensionadas mediante la función "resize".

Para el test, utilizamos un total de 864, de las cuales 560 son positivas y 304 son negativas.

Obtenemos la siguiente matriz de confusión:

	COCHE	NO-COCHE
COCHE	512	48
NO-COCHE	80	224

Tabla 1: Matriz de confusión de la prueba 1.

A partir de estos valores podemos calcular los diferentes parámetros vistos anteriormente.

EXACTITUD	PRECISIÓN	SENSIBILIDAD	ESPECIFICIDAD	TRUE POSITIVE RATE	FALSE POSITIVE RATE
0.852	0.865	0.914	0.737	0.914	0.263

Tabla 2: Resultados de la prueba 1.

PRUEBA 2:

Para la realización de la segunda prueba, primero se aplicarán las técnicas de bootstrapping y aprendizaje activo con el conjunto de imágenes de refino, que corresponde aproximadamente al 20% de las imágenes. Cuando dejemos de obtener falsos positivos y falsos negativos en nuestro clasificador, volveremos a cargar el clasificador ya realimentado al conjunto de imágenes de test.

En este caso, todas las imágenes también han sido redimensionadas mediante la función "resize".

En primer lugar vamos a aplicar el clasificador al conjunto de refino para observar los resultados. Este conjunto de imágenes está formado por 2623, de las cuales 1660 son positivos y 963 son negativos.

Obtenemos la siguiente matriz de confusión:

	COCHE	NO-COCHE
COCHE	1434	226
NO-COCHE	195	768

Tabla 3: Matriz de confusión del conjunto de refino de la prueba 2.

Ahora calculamos los distintos parámetros:

EXACTITUD	PRECISIÓN	SENSIBILIDAD	ESPECIFICIDAD	TRUE POSITIVE RATE	FALSE POSITIVE RATE
0.839	0.880	0.864	0.797	0.864	0.203

Tabla 4: Resultados del conjunto de refino de la prueba 2.

Una vez obtenidos estos resultados, se procede a realimentar el clasificador, de manera que los falsos positivos obtenidos se introducen en el conjunto de entrenamiento de negativos, mientras que los falsos negativos obtenidos se introducen en el conjunto de entrenamiento de positivos.

Posteriormente se vuelve a entrenar y se obtiene un nuevo clasificador, que se aplicará al conjunto de test, obteniendo los siguientes resultados:

	COCHE	NO-COCHE
COCHE	541	19
NO-COCHE	52	252

Tabla 5: Matriz de confusión del conjunto de test de la prueba 2 aplicando la realimentación.

EXACTITUD	PRECISIÓN	SENSIBILIDAD	ESPECIFICIDAD	TRUE POSITIVE RATE	FALSE POSITIVE RATE
0.918	0.912	0.967	0.829	0.967	0.171

Tabla 6: Resultados del conjunto de test de la prueba 2 aplicando la realimentación.

PRUEBA 3:

Para la realización de la prueba 3 se han tomado todas las imágenes redimensionadas mediante el script de Python explicado anteriormente.

Una vez que obtenemos el nuevo clasificador, utilizando un total de 3554 imágenes, de las cuales 2317 corresponden a imágenes positivas y 1237 corresponden a imágenes negativas. Volvemos a probar a predecir con el conjunto de imágenes de test que, en este caso se utilizarán un total de 844 imágenes, de las cuales 521 son positivas y 323 son negativas.

Se obtienen los siguientes valores:

	COCHE	NO-COCHE
COCHE	503	18
NO-COCHE	44	279

Tabla 7: Matriz de confusión de la prueba 3.

EXACTITUD	PRECISIÓN	SENSIBILIDAD	ESPECIFICIDAD	TRUE POSITIVE RATE	FALSE POSITIVE RATE
0.926	0.920	0.965	0.864	0.965	0.136

Tabla 8: Resultados de la prueba 3.

PRUEBA 4:

En la realización de la prueba 4 también se han empleado imágenes redimensionadas mediante el script de Python.

En este caso, una vez que obtenemos el clasificador, se realiza la predicción al conjunto de refino, para poder aplicar las técnicas de bootstrapping y aprendizaje activo al clasificador, como se ha hecho en la prueba 2.

Este conjunto de refino está formado por 1211 imágenes, de las cuales 788 son positivas y 423 son negativas.

Obtenemos los siguientes resultados:

	COCHE	NO-COCHE
COCHE	751	37
NO-COCHE	63	360

Tabla 9: Matriz de confusión del conjunto de refino de la prueba 4.

EXACTITUD	PRECISIÓN	SENSIBILIDAD	ESPECIFICIDAD	TRUE POSITIVE RATE	FALSE POSITIVE RATE
0.917	0.923	0.953	0.851	0.953	0.149

Tabla 10: Resultados del conjunto de refino de la prueba 4.

Después de comprobar los resultados, se realiza la realimentación del clasificador, de manera similar a como se ha hecho en la prueba 2.

Acto seguido se vuelve a obtener un nuevo clasificador, que se va a aplicar al conjunto de imágenes de test, obteniéndose los siguientes resultados:

	COCHE	NO-COCHE
COCHE	516	5
NO-COCHE	21	302

Tabla 11: Matriz de confusión del conjunto de test de la prueba 4 aplicando la realimentación.

EXACTITUD	PRECISIÓN	SENSIBILIDAD	ESPECIFICIDAD	TRUE POSITIVE RATE	FALSE POSITIVE RATE
0.970	0.961	0.990	0.935	0.990	0.065

Tabla 12: Resultados del conjunto de test de la prueba 4 aplicando la realimentación.

Finalmente, en la siguiente tabla se muestra una comparativa con los resultados de todas las pruebas realizadas al conjunto de imágenes de test:

PRUEBAS	Nº de imágenes de entrenamiento	Nº de imágenes de entrenamiento positivas	Nº de imágenes de entrenamiento negativas	Nº de imágenes de test	Nº de imágenes de test positivas	Nº de imágenes de test negativas
PRUEBA 1	5999	3352	2247	864	560	304
PRUEBA 2	5999	3352	2247	864	560	304
PRUEBA 3	3554	2317	1237	844	521	323
PRUEBA 4	3554	2317	1237	844	521	323

Tabla 13: Número de imágenes utilizadas en cada prueba.

PRUEBAS	EXACTITUD	PRECISIÓN	SENSIBILIDAD	ESPECIFICIDAD	TRUE POSITIVE RATE	FALSE POSITIVE RATE
PRUEBA 1	0.852	0.865	0.914	0.737	0.914	0.263
PRUEBA 2	0.918	0.912	0.967	0.829	0.967	0.171
PRUEBA 3	0.926	0.920	0.965	0.864	0.965	0.136
PRUEBA 4	0.970	0.961	0.990	0.935	0.990	0.065

Tabla 14: Comparativa de los resultados de todas las pruebas realizadas.

Como se aprecia en la tabla 14, los resultados de la prueba 4 han sido los más satisfactorios.

6. TRABAJOS FUTUROS.

En este capítulo se va a comentar algunas ideas de futuro que se podrían aplicar para mejorar y complementar el funcionamiento del algoritmo.

En primer lugar, lo más importante sería integrar dicho sistema en un vehículo comercial y, a partir de ahí, poder desarrollar otras mejoras como se mencionan a continuación:

- Detección de otros obstáculos que aparecen en la vía urbana aparte de coches, como por ejemplo, peatones, motos, bicicletas, etc.
- Mejorar la detección de los obstáculos en condiciones de mala visibilidad. Para ello se puede dotar al sistema de una cámara infrarroja, ya que estas cámaras no dependen en gran medida de la iluminación. Habría que realizar una combinación del sistema de detección mediante cámaras normales y del sistema de detección mediante cámaras infrarrojas, para que funcionase correctamente en cualquier situación.
- Una vez conocida las posiciones reales de los distintos obstáculos puede aplicarse una técnica de seguimiento para predecir las trayectorias de estos obstáculos, sobre todo de peatones. Esto permitiría estudiar si alguna de las posibles trayectorias coincide con la del vehículo y de ser así, alertar al conductor con la antelación adecuada para que reaccione ante esa situación. Para este fin se puede recurrir a la aplicación del filtro Kalman que permite obtener las posibles trayectorias de los obstáculos localizados en la imagen mediante estimaciones de los estados de un sistema lineal.
- Realizar un sistema de alerta para al conductor en el caso que se detecte un obstáculo cercano al coche.
- Dotar al vehículo de un sistema de frenado de emergencia en caso de que se encuentre un obstáculo cercano al coche y el conductor no haya reaccionado a tiempo.

7. CONCLUSIONES.

En el presente proyecto, se ha llevado a cabo el diseño de un algoritmo para la detección de obstáculos en un entorno urbano.

En primer lugar, respecto al etiquetado de imágenes, en las primeras secuencias de imágenes se empezó empleando el primer método de etiquetado, aunque finalmente fue el segundo método el más utilizado para este propósito, ya que es un método mucho más rápido y, al tener tantas imágenes, resulta más fácil de obtener las ROI.

En relación a los resultados obtenidos, cabe destacar que son bastante buenos en todas las pruebas realizadas. Estos resultados, aunque teóricamente son buenos, no representan exactamente el funcionamiento del algoritmo, ya que para la obtención del clasificador se ha trabajado con un conjunto de imágenes no demasiado numeroso, ya que no disponíamos de más secuencias de imágenes. Además, a este problema hay que añadirle que muchas de las imágenes del conjunto son muy parecidas, por lo que el clasificador aprende muy bien las características de esas imágenes pero luego en la realidad van a aparecer situaciones muy distintas. Lo ideal habría sido poder contar con un mayor número de secuencias de imágenes totalmente distintas y grabadas en distintos lugares.

En cualquier caso, si analizamos los resultados obtenidos, podemos deducir que se mejoran los resultados si realizamos la realimentación al clasificador. Esto se debe a que al realizar la realimentación, se eliminan los falsos positivos y los falsos negativos que aparecían en el clasificador, por lo que estamos mejorando dicho clasificador.

Otra cosa a destacar es que los resultados también mejoran si utilizamos el conjunto de imágenes redimensionadas mediante el script de Python. Esto se debe a que, mediante esta forma de redimensionado, no se distorsionan las imágenes. El principal problema reside en que algunas imágenes no se redimensionan correctamente y tienen que ser eliminadas manualmente del conjunto de imágenes, por lo que, aparte de ser un proceso algo costoso de realizar, se emplea un menor número de imágenes para el entrenamiento.

Se puede concluir en que el desarrollo del algoritmo ha sido satisfactorio, aunque se podría mejorar varios aspectos como se han comentado.

8. BIBLIOGRAFÍA.

1. DGT, *"Las principales vías de la siniestralidad en España 2013"*, http://www.dgt.es/Galerias/seguridad-vial/estadisticas-e-indicadores/publicaciones/accidentes-urban/Siniestralidad_Urbana_2013_EE.pdf
2. DGT, *"Se consolida el descenso sostenido de la siniestralidad desde 2003 en vías interurbanas"* , http://www.dgt.es/es/prensa/notas-de-prensa/2015/20150105-mir_BALANCE-2014_seguridad-vial_nota.shtml
3. CEA, *"Seguridad activa y pasiva del vehículo"*, http://www.cea-online.es/reportajes/seguridad.asp
4. Felipe Jiménez Alonso, "*Integración de diferentes sistemas ADAS en un interfaz de usuario adaptado a las características del conductor"*, http://oa.upm.es/7568/1/INVE_MEM_2010_77369.pdf
5. Ministerio de Economía y Competitividad, *"Sistemas avanzados de asistencia al conductor"*, http://www.investinspain.org/invest/wcm/idc/groups/public/documents/documento/mda0/ntez/~edisp/4513378.pdf
6. Freescale, " *Advanced driver assistance systems (ADAS)"*, http://www.freescale.com/applications/automotive/adas:ADAS
7. EuroNCAP Advanced Systems, http://www.euroncap.com/es/seguridad-en-los-veh%C3%ADculos/descripci%C3%B3n-de-los-premios/
8. Noticias coches, *"¿Que es el control de velocidad adaptativo?"*, http://noticias.coches.com/noticias-motor/que-es-el-control-de-velocidad-adaptativo/3833
9. Km77, http://www.km77.com/glosario/a/alerta.asp
10. Circula seguro, *"Sistemas para evitar el ángulo muerto en los espejos"*, http://www.circulaseguro.com/sistemas-para-evitar-el-angulo-muerto/
11. Km77, "*sistema de visión nocturna"*, http://www.km77.com/glosario/v/visionnoct.asp
12. Motorpasion, "*¿son fiables los sistemas de visión nocturna?"*, http://www.motorpasion.com/seguridad/son-fiables-los-sistemas-de-vision-nocturna
13. Seguridad Vial en la Empresa, "*Dispositivos para el coche: detección de la fatiga y distracción al volante"*, http://www.seguridadvialenlaempresa.com/seguridad-empresas/actualidad/noticias/dispositivos-para-el-coche-deteccion-de-la-fatiga-y-distraccion-al-volante.jsp
14. Circula Seguro, "*Sistemas de detección de la fatiga y falta de concentración al volante*", http://www.circulaseguro.com/sistemas-de-deteccion-de-la-fatiga-y-falta-de-concentracion-al-volante/

15. Noticias coches, "*Cámaras que leen señales de tráfico*", http://noticias.coches.com/noticias-motor/camaras-que-leen-las-senales-de-trafico/30755
16. Circula Seguro, "*¿Qué es el TPMS o control de presión de los neumáticos?*", http://www.circulaseguro.com/que-es-el-tpms-o-control-de-presion-de-los-neumaticos/
17. Audi, "*Sistemas de iluminación. Luz adaptativa*", http://www.audi.com.sv/aola/brand/es_sv/eficiencia/efficiency_technologies/light_systems/adaptive_lighting_control.html
18. Volkswagen, "*Sistemas de iluminación*", http://www.volkswagen.co/mundo_volkswagen/innovacion_y_tecnologia/carroceria/sistemas_de_iluminacion
19. Xataka, "*Tecnología para el coche: sistema de aparcamiento automático*", http://www.xataka.com/automovil/tecnologia-para-el-coche-sistemas-de-aparcamiento-automatico
20. Circula Seguro, *"¿Qué es la comunicación entre vehículos?"*, http://www.circulaseguro.com/que-es-la-comunicacion-entre-vehiculos/
21. CEA Seguridad Vial, "*¿Qué es eCall?*", http://www.seguridad-vial.net/articulos/120-que-es-el-ecall
22. Motorpasion futuro, "*Como funciona el coche autónomo de google*", http://www.motorpasionfuturo.com/coches-del-futuro/como-funciona-el-coche-autonomo-de-google
23. Boost, http://www.boost.org/
24. ComunidadQ, *"Administrar el Sistema de archivos con C++ y Boost"*, http://www.comunidadq.com/articulos/ver/3/administrar-el-sistema-de-archivos-con-c-y-boost
25. Sqlite, "*About Sqlite*", https://www.sqlite.org/about.html
26. OpenCV, http://opencv.org/
27. Laboratorio de sistemas inteligentes, http://portal.uc3m.es/portal/page/portal/dpto_ing_sistemas_automatica/investigacion/lab_sist_inteligentes
28. F. García , A. Ponz, D. Martín, J.M Armingol y A. de la Escalera, "*Fusión de Escáner Laser y Visión por Computador para la Detección de Peatones en Entornos Viarios*", http://www.sciencedirect.com/science/article/pii/S1697791215000126
29. Ayúdanos a conocer, "*Histogramas de gradientes orientados*", http://ayudamosconocer.com/significados/letra-h/histograma-de-gradientes-orientados.php
30. Grace Tsai, " *Histogram of oriented gradients*", http://web.eecs.umich.edu/~silvio/teaching/EECS598_2010/slides/09_28_Grace.pdf
31. Jon Intxaurbe Txarterina, "*Detección de personas*", https://addi.ehu.es/bitstream/10810/13345/2/main.pdf

32. OpenCV, "*Introduction to Support Vector Machines",* http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html?highlight=svm
33. Betancur, Gustavo A, *"Las Máquinas de soporte vectorial (SVMs)",* http://www.redalyc.org/articulo.oa?id=84911698014
34. Scikit Learn, *"Support Vector Machines",* http://scikit-learn.org/stable/modules/svm.html
35. Universidad de Sevilla, *"Detección Multiusuario para DS-CDMA basado en SVM",* http://bibing.us.es/proyectos/abreproy/11185/fichero/Volumen+1_Detector+Multiusuario+para+DS-CDMA+basado+en+SVM%252F7.+Support+Vector+Machines%252FSupport+Vector+Machines.pdf
36. StackOverflow, http://stackoverflow.com/
37. C++ con Clase, http://www.c.conclase.net/
38. Gary Bradski and Adrian Kaehler, *"Learning OpenCV"*
39. Javier García de Jalón, José Ignacio Rodríguez, José María Sarriegui, Alfonso Brazález, *"Aprenda C++ como si estuviera en primero"*
40. Sebest, "Ejemplos básicos de OpenCV", http://www.sebest.com.ar/?q=node/79
41. OpenCV Tutorial C++, *"How to detect mouse clicks and moves?"*, http://opencv-srf.blogspot.com.es/2011/11/mouse-events.html
42. Learn OpenCV by examples, "*Detect mouse clicks and moves on image window"*, http://opencvexamples.blogspot.com/2014/01/detect-mouse-clicks-and-moves-on-image.html
43. StackOverflow, *"How to train a SVM with OpenCV based on a set of images?"*, http://stackoverflow.com/questions/16876960/how-to-train-an-svm-with-opencv-based-on-a-set-of-images
44. Github, *"Train HOG"*, https://github.com/DaHoC/trainHOG/blob/master/main.cpp
45. StackOverflow, "*How to load previously stored SVM classifier?",* http://stackoverflow.com/questions/24135372/how-to-load-previously-stored-svm-classifier
46. StackOverflow, "*How to use HOGDescriptor::detectMultiscale() with custom SVM?"*, http://stackoverflow.com/questions/22159892/opencv-how-to-use-hogdescriptordetectmultiscale-with-custom-svm
47. Oldemar Rodríguez, *"Aprendizaje supervisado"*, http://oldemarrodriguez.com/yahoo_site_admin/assets/docs/Presentaci%C3%B3n_-_KNN.20085205.pdf